2020-12

INTEGRATING DEVOPS INTO NAVY COMBAT SYSTEMS DEVELOPMENT

Miller, Andrew W.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INTEGRATING DEVOPS INTO NAVY COMBAT SYSTEMS DEVELOPMENT

by

Andrew W. Miller

December 2020

Thesis Advisor:	Ronald E. Giachetti
Second Reader:	Douglas L. Van Bossuyt

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE INTEGRATING DEVOPS INTO NAVY COMBAT SYSTEMS DEVELOPMENT	**5. FUNDING NUMBERS**
6. AUTHOR(S) Andrew W. Miller	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis seeks to answer three questions concerning the Navy's adoption of DevOps and its practices. Those questions are: What is DevOps in a naval context? What stands in the way of that adoption? What are some ways that the Navy can overcome those obstacles? By drawing upon both an extensive review of literature on the topic, as well as interviews with subject-matter experts, this work provides a comprehensive understanding of the breadth and complexity of the change needed in order for the Navy to adopt a culture of DevOps as well as its attendant practices. Pursuant to the same end, this thesis proposes process architectures for continuous integration, continuous testing, and continuous certification, as well as the reorganization of the Navy's combat systems development hierarchy necessary for the transition to DevOps.

14. SUBJECT TERMS DevOps, software development, AGILE, systems engineering, critical success factors, requirements analysis, DevSecOps, cyber-physical systems	**15. NUMBER OF PAGES** 127
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

INTEGRATING DEVOPS INTO NAVY COMBAT SYSTEMS DEVELOPMENT

Andrew W. Miller
Lieutenant, United States Navy
BS, Virginia Military Institute, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2020

Approved by: Ronald E. Giachetti
Advisor

Douglas L. Van Bossuyt
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis seeks to answer three questions concerning the Navy's adoption of DevOps and its practices. Those questions are: What is DevOps in a naval context? What stands in the way of that adoption? What are some ways that the Navy can overcome those obstacles? By drawing upon both an extensive review of literature on the topic, as well as interviews with subject-matter experts, this work provides a comprehensive understanding of the breadth and complexity of the change needed in order for the Navy to adopt a culture of DevOps as well as its attendant practices. Pursuant to the same end, this thesis proposes process architectures for continuous integration, continuous testing, and continuous certification, as well as the reorganization of the Navy's combat systems development hierarchy necessary for the transition to DevOps.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AO	authorizing official
APB	Advance Processor Build
ASA	Adaptive Squad Architecture
ASN RDA	Assistant Secretary of the Navy for Research, Development, and Acquisitions
ATO	authority to operate
C2C24	Compile to Combat - 24 Hours
C4ISR	command, control, computers, communications, computers, intelligence, surveillance, and reconnaissance
CA	Certification Authority
CBA	capabilities based assessment
CDD	Capabilities Description Document
COTF	Commander, Operational Test and Evaluation Force
CPCR	Computer Program Change Request
CSSQT	Combat Systems Ship Qualification Trials
DAS	Defense Acquisitions System
DAU	Defense Acquisitions University
DevOps	development and operations
DIACAP	Department of Defense (DoD) Information Assurance (IA) Certification and Accreditation Process
DoD	Department of Defense

DoN	Department of the Navy
FAR	Federal Acquisitions Regulation
FFP	firm-fixed-price
HRE	highly regulated environment
HRO	high reliability organization
IA	Information Assurance
IaaS	Infrastructure as a Service
ICD	Initial Capabilities Document
IDIQ	indefinite quantity/indefinite delivery
IOC	initial operational capability
IPT	Integrated Product Team
IT	information technology
IWS	Integrated Weapons Systems
JCIDS	Joint Capabilities Integration and Development System
JROC	Joint Requirements Oversight Council
MAGTF	Marine Air-Ground Task Force
MASINT	measurement and signature intelligence
MBSE	model based systems engineering
MUX	Marine Air-Ground Task Force (MAGTF) unmanned aircraft system (UAS) Expeditionary
NAVAIR	Naval Air Systems Command
NAVSEA	Naval Sea Systems Command

NAVWAR Naval Information Warfare Systems Command

NPS Naval Postgraduate School

OPNAV Office of the Chief of Naval Operations

PA Programmatic Authority

PEO Program Executive Office

PEO C4ISR Program Executive Office (PEO) command, control, computers, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR)

PEO EIS PEO Enterprise Information Systems

PEO IWS PEO Integrated Weapons Systems (IWS)

PPBE Planning, Programming, Budget, and Execution

POE planned operating environment

RMF Risk Management Framework

ROC required operating capabilities

SAFe Scaled Agile Framework

SOVT Systems Operation and Verification Test

SRWBR short range wide band radio

SSDS Ship Self-Defense System

SYSCOM systems command

TA Technical Authority

TCP Transmission Control Protocol

TEMP test and evaluation master plan

TOR	Test Observation Report
TR	Trouble Report
TRL	technology readiness level
TYCOM	Type Commander
UAS	unmanned aircraft system
UDP	User Datagram Protocol
USG	United States government
USN	U.S. Navy
VPN	virtual private network

Executive Summary

Over the past half-decade, Navy acquisitions programs have come under attack from defense officials, members of the armed forces, and Congress for their continual cost overruns and glacial pace of delivery. This has prompted the Navy to look for new ways of doing business in order to speed up the development and delivery of combat systems. As part of these efforts to modernize, the Navy has sought to adopt best practices from private industry and Silicon Valley. These practices include Agile software development and DevOps, which is the combination of software development and IT operations into one continuous whole. This DevOps concept seeks to bring developers and operators into a harmonious relationship to improve communication, increase development speed, and reduce the rate of errors and inefficiencies in the implementation of new technology.

Implementing DevOps in the Navy is a challenge that stretches across the entirety of the Naval Service and the key to implementing these changes successfully is to have a solid understanding of the topics involved, a common vernacular with which to discuss the implementation of these processes, and a framework of requirements from which to build the acquisitions and maintenance life cycle for each system involved. Many of these challenges stem from a deeply-entrenched culture within the Naval Service as a whole and the Navy's Acquisitions Corps specifically. This culture is resistant to change and is wary of risks and will take substantive work to transition the Navy's development processes from a linear "waterfall" methodology to the continuous cyclic processes that make up DevOps as shown in Figure 1.

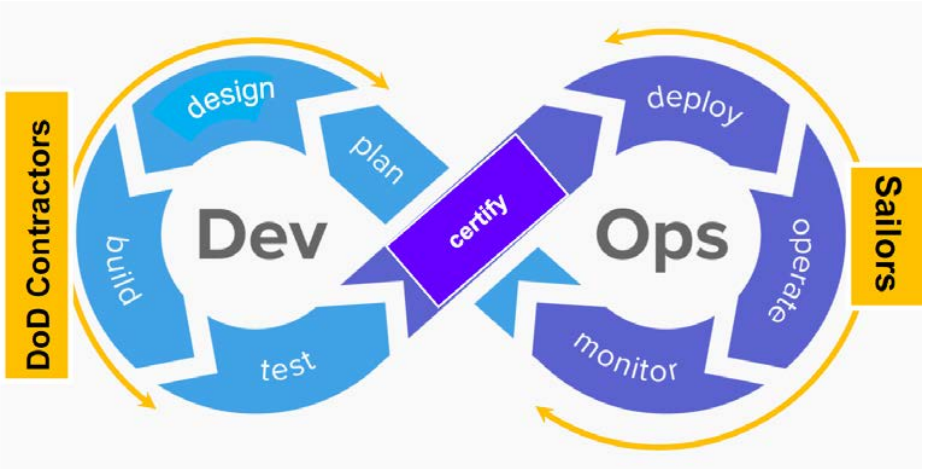

Figure 1. Proposed Navy DevOps Cycle. Adapted from [1].

Under this new paradigm, the planning, development, testing, and deployment of new combat systems becomes continuous and nearly infinite. Like Agile Development, this will require a focus on the end user of each combat system which is the Sailor at sea. Developers and engineers will directly incorporate the operational feedback from the fleet into the development and revision of systems designs that will then be tested, certified, and sent back out to the fleet. This feedback loop is the lynch pin of any DevOps system and what differentiates DevOps from Agile.

To achieve this state of continuous feedback, development, and delivery, the Navy will need to change the way it approaches combat system testing and certification. It will also need to build the necessary physical and digital infrastructure to facilitate this new acquisitions system. This work will result in the creation of continuous certification, continuous testing, and continuous integration cycles that will speed the development and fielding of new combat capabilities on board ships and submarines. This will better equip Sailors to fight the wars of the future and win.

References:

[1] Z. Banach, "What is DevSecOps," Oct 2019. Available: https://www.netsparker.com/blog/web-security/what-is-devsecops/

Acknowledgments

I would like to thank my parents, Randy and Karen Miller, for their continued support throughout my Navy career. I would also like to thank my daughter, Susan, for being so patient with me as the Navy has kept her daddy away from her for so long. I'd also like to extend my thanks to my advisors Doctors Giachetti and Van Bossuyt for their mentorship and long-suffering patience with me during the construction of this work.

The following people have been instrumental in the process of creating my thesis. Their dedication and donation of their time and experience has been in invaluable in my research and understanding of the topics herein. Without their assistance, none of this would have been possible.

- RDML Peter Williams, USN (ret.)
- Mr. Richard Jack
- CAPT Dave Markert, USN (ret.)
- CDR Jason Hunter, USN
- CDR Terry Johnson, USN (ret.)
- CDR David Parsons, USN (ret.)
- LCDR Timothy Beach, USN
- CPT Sean McIntosh, USA
- 2LT Kyle Taylor, USAF
- Mr. Socrates Frangis
- Mrs. Lori Skowronski

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Over the past half-decade, the Defense Acquisitions System (DAS) has come under attack from defense officials, members of the armed forces, and Congress for its continual cost overruns and glacial pace of delivery [1]. This has prompted the Department of Defense (DoD) to look for new ways of doing business and to speed up the development and delivery of military systems. As part of these efforts to modernize, the DoD has sought to adopt to the best practices from private industry and Silicon Valley [2]. These practices include Agile software development and development and operations (DevOps), which is the combination of software development and IT operations [3]. This DevOps concept seeks to bring developers and operators into a harmonious relationship to improve communication, increase development speed, and reduce the rate of errors and inefficiencies in the implementation of new technology.

With the implementation of DevOps, the Navy seeks to create an environment of continuous development, integration, testing, and delivery. This concept is at the heart of DevOps as it seeks to treat the software and hardware systems as living organisms that need constant care and attention [4]. This means that software and hardware development is happening alongside systems integration, which is occurring at the same time as systems testing. Moreover, while those tasks are being performed, an approved and certified version of the system or one of its components is being fielded or installed on a ship for use by Sailors.

This is a large departure from the traditional ways that the Navy performs systems development and deployment. Other services, such as the Air Force [5], have already made strident efforts to put these processes and philosophies into practice [6]. However, these efforts are still in their most nascent of stages. As of yet, no single team has been able to determine how best to implement a DevOps model for combat systems.

1.1 Competitive Advantage

Prompting this sea change in development culture is the new era of great power competition the nation finds itself in. This renewed competition with other peer and near-peer world powers has necessitated the need to field more modern and more capable combat systems than those of our adversaries and on a shorter time table because the U.S. fears it is losing its competitive advantage due to technical superiority. These combat systems include the computers, communications systems, missiles, guns, and all manner of electronic warfare systems found on board the Navy's ships, submarines, and aircraft. Commensurate with this need to modernize their systems and expand the capabilities of the Fleet, the Navy must innovate faster and more effectively [7].

As the U.S. finds itself facing a potential conflict with nations such as Russia or China within the next decade [8], it must arm itself with the best combat systems it possibly can, using the latest technology. This race to adapt to new technologies is one of the main drivers of organizational development and change identified by Hendrick in [9]. Hendrick argues that as technology changes, the way that human beings interact with the machines they use to accomplish tasks also changes. They also change the way that humans interact with each other, as technology facilitates new ways to communicate (video teleconferencing, for instance) and share information. A shining example of this change in communication is the rapid shift to telework and distance learning caused by the COVID-19 outbreak in Spring of 2020. When placed in the greater context of a great power competition, it becomes evident that this technological race is not simply about technology but also about world competition, which Hendrick identified as the second main driver of change.

The forces driving change internally within the Navy are the Navy's leadership who are reacting to the pressure from perceived future threats to the U.S. and the Navy's ability to project power and influence around the world. Seeing the rapid pace of technological development within the IT industry and in places such as Silicon Valley, the admirals and civil service leaders in charge of the Navy's acquisitions process, seek to borrow the "keys to success" from who they perceive to be the nation's leading innovators.

This desire to adopt best practices from industry seeks to remedy problems that have plagued the Navy's acquisitions programs since the 1990s. These hurdles include glacial systems development, degraded operational capabilities once the systems are fielded, and

budget overruns [10]. Currently, combat systems take anywhere from 18 months to 8 years to be developed and then fielded [11], and the Navy's leadership seek to reduce this to a matter of days [12]. Gibson et al. explain that this desire to correct shortfalls, such as those experienced by the Navy's Acquisitions Corps, is one of the strongest motivators for change once these shortfalls are recognized [13].

1.2 The Current Acquisitions Process Cannot Keep Pace

Unfortunately, the current acquisitions system of the DoD is firmly stuck in the era of the Cold War and is slow to change and adopt new ideas. Called the DAS, the DoD's system of acquisitions processes is colloquially referred to as the "waterfall" model due to its rigid linear construction with a definite start point and end point built in [14]. The DAS model places inflexible program milestones and design reviews into the development process as illustrated in Figure 1.1.

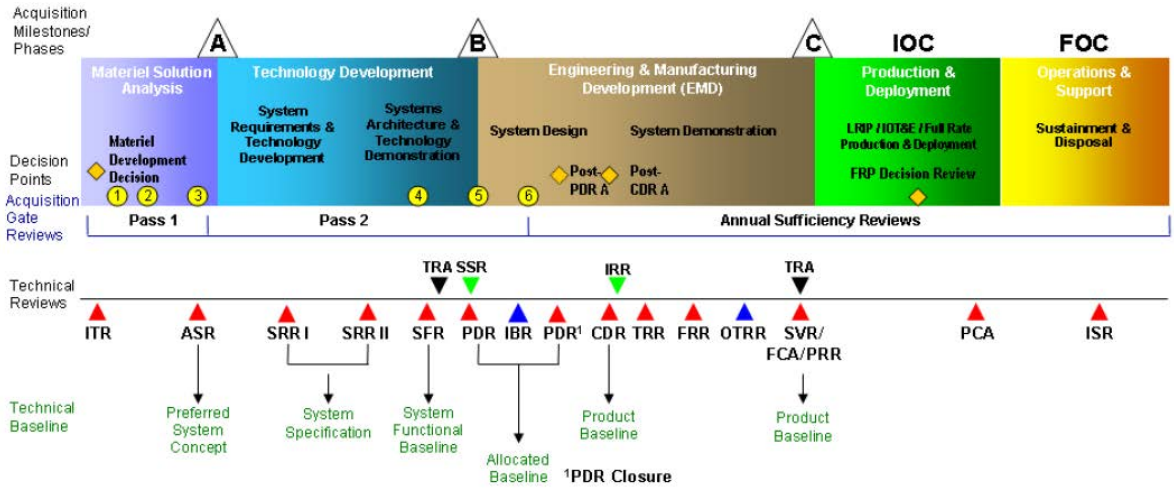

Figure 1.1. Current Acquisitions Process for Naval Combat Systems. Source: [14].

These design reviews are predicated on certain development milestones being met. In Figure 1.1, these are labeled as the letters A, B, and C in triangles. These milestones mandate that certain levels of systems maturity must be achieved and a certain amount of administrative review and systems testing be conducted. Because of this, each milestone requires some

sort of system baseline configuration (as noted in green text in 1.1) that lock in place all further progress in technological development for that system [14]. To make changes to these baseline configurations after they have been approved requires multiple formal design reviews and updates to the Capabilities Description Document (CDD) that cause delays in system development and delivery [15].

By creating these baselines, the Navy is able to create a single version of a system that can be tested and certified for installation aboard ships and submarines. This certification process is designed to ensure that any combat system being issued to the Fleet, is able to perform its functions safely, reliably, and according to the specification that was promised by its developer. Unfortunately, this forces the need to restart the entire DAS process from the beginning every time something new is to be added to a system or a new capability developed. Even small changes in software code requires over a year of testing and investigation before it can be delivered to the Fleet [11]. This often results in technology being obsolete by the time it reaches the hands of Sailors. Moreover, should the Navy find itself in combat with a peer adversary, it may need to delivery new technologies to the fleet in a matter of days or weeks, and not years.

Further exacerbating this lethargic pace is the current organization of the defense acquisition community into a hierarchical structure that separates the DoD contractors designing and building the combat systems, the Programmatic Authority (PA) overseeing acquisitions process, the Technical Authority (TA) certifying the combat systems, and the Sailors operating and maintaining the combat systems [14]. Part of this separation is simply due to the Navy's reliance on strict hierarchies for maximum combat effectiveness during war, but also because Congressional statutes require that distinct lines of authority to prevent conflicts of interest. This strict hierarchical structure was put into place during the 20th century for the management of the developing and testing of hardware systems [16]. This was when the systems the Navy and DoD acquired were mainly reliant upon closed hardware systems that needed entire ecosystems of proprietary components and rudimentary analog or mechanical control software.

For 21st century software-based systems and their rapid pace of technological advancement, the current DAS model seems antiquated and inadequate. Modern combat systems rely upon complex digital control software and operating systems. Modern development practices of

these combat systems also require tight integration of and close collaboration between all developers, engineers, testers, and operators of those systems. These modern methods are referred to as being "Agile," which is a philosophy of iterative software design, development, and delivery [17]. So as the Navy becomes more and more reliant upon software and its symbiotic relationship with hardware, the Navy's acquisitions processes must adapt to the way that modern software is developed.

1.3 Purpose of This Thesis

This thesis will document the work necessary for the implementation of DevOps practices within the Navy's acquisitions community. Through the following sections, this work will define a common definition of DevOps for the Navy, examine the challenges impeding implementation of DevOps, propose acquisitions and organizational models to overcome the most intractable of these challenges, and then discuss future work that must be accomplished. Sources will be drawn from a literature review as well as interviews with industry experts, Navy program management personnel, and DoD contractors. Qualitative analysis will then be used to create models for the acquisitions, development, and testing of Navy combat systems. It is the goal of this work to provide the Navy's acquisitions professionals with the necessary background and programmatic information necessary to increase the speed of acquisitions.

To accomplish this goal, this thesis will be divided into four sections, each addressing some aspect of the transition to DevOps. Chapter 2 discusses what DevOps is and provides context to how the Navy should define it. Chapters 3 and 4 discuss the challenges that stand in the way of the Navy's adoption of DevOps and proposes solutions to overcome those challenges, respectively. Chapter 5 will then explain what future work will need to be completed to lay the ground work necessary for the Navy to fully transition to DevOps.

1.4 Methodology

To accomplish the goal of performing a holistic investigation of the challenges and solutions confronting the Navy in its adoption of DevOps, a comprehensive review of available literature was conducted. To understand the challenges and issues unique to the Navy, semi-

structured interviews were conducted with experts capturing quotes and thoughts regarding the field of DevOps, Agile development, and Navy combat systems development that will be included throughout. These quotes and thoughts are lifted verbatim or paraphrased from interviews and personal correspondence with these experts. In total, ten interviews were conducted that lasted anywhere from forty-five minutes to two hours and consisted of five to six complex starting questions concerning the technical, cultural, regulatory, and process challenges that stand before the Navy in its attempt to modernize its development programs. These questions asked the respondents to draw upon their professional experiences, current work in the field, and knowledge of both the Navy's acquisitions programs as well as those in private industry. The interviews were conducted in a semi-structured manner in order to stimulate conversation and provide a forum for each expert to vocalize their full opinions, experiences, and expertise on subjects germane to the Navy's adoption and implementation of DevOps. Table 1.1 lists the expert's who were intereviewed with their position and brief description of their expertise.

Table 1.1. DevOps Experts

Expert	DevOps Experience
Department of the Navy (DoN) contractor at Naval Air Systems Command	Over 30 years as a Naval Officer and 13+ years working as a contractor and Agile consultant.
Senior Software Engineer at Naval Information Warfare Systems Command	Over 25 years working with Navy information technology (IT) systems and championing DevOps and Agile practices.
Chief Engineer at DoD contractor	Over 20 years as a Naval Officer and 15+ years as a software developer implementing Agile and DevOps practices.
Project Manager at PEO Integrated Weapons Systems	Over 15 years as an officer in the Naval Reserve and is employed doing IT development using DevOps and Agile practices in civilian life.
DoN contractor at PEO Integrated Weapons Systems	Over 40 years as a Naval Officer and civilian Acquisitions Professional with a focus on combat systems certification and testing.
Program Manager at Naval Air Systems Command	Over 20 years as a Naval Officer and Acquisitions Professional with a background in rapid prototyping and Agile development.
Senior Software Engineer at PEO Integrated Weapons Systems	Over 15 years developing and testing Naval weapons and cyber systems.
Assistant Program Manager at PEO Integrated Weapons Systems	Over 10 years as a Naval Officer and Acquisitions Professional.
Assistant Program Manager at PEO Soldier	Over 15 years of enlisted and commissioned experience in the Army and a background in Agile development.
Scrum Master at Air Force's Kessel Run Program Office	Over 10 years enlisted and commissioned in the USAF and formal education in IT systems.
Systems Certification Manager at PEO Integrated Weapons Systems	Over 10 years as a DoN civilian.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
What Is DevOps?

To modernize their acquisitions process, the DoD and the DoN have sought to borrow the best practices from private industry. Specifically, they have strived since 2018 to adopt a continuous development model called DevOps. This new model, as illustrated in 2.1, creates a work environment where development (Dev), testing, and operations (Ops) are part of a single infinite cycle that feed into each other. Contrast this with the "waterfall" DAS model in Figure 1.1 where each of these phases of the acquisitions process are segmented with concrete start and end dates.

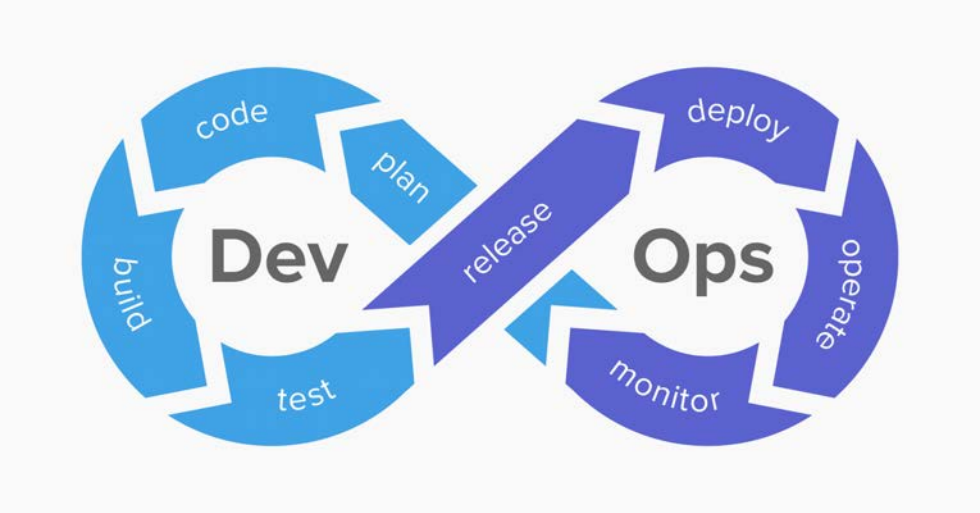

Figure 2.1. Typical DevOps Cycle. Source: [18].

In this new DevOps acquisitions system shown in Figure 2.1, the developers will feed a product to the testers who will certify that it is safe, reliable, and capable of meeting an operational need (as defined by the Navy's operational leadership and the program office). These testers will pass testing results back to the developers in near real time so that data can be used to make improvements and to fix shortfalls within the software or hardware. With these improvements made, this new change or fix is deployed by the operators for use by the end users. In the case of the Navy, the operators and users are Sailors stationed

on ships or submarines who both operate and maintain combat systems. After using the combat systems, Sailors and their commanders will then provide feedback to the developers and testers so that they can improve the combat system and the testing methods used to certify it for operation. Unlike DAS, this cycle happens without an end and without stopping throughout the life of the program.

But what exactly is DevOps anyway? This is a question that has plagued the Navy and even the experts interviewed. In most of the interviews, the experts wondered what DevOps meant or even looked like in a naval context. In fact, this confusion over the precise definition of DevOps for the Navy was the impetus for this thesis. In conversations that the author had with leaders in the Navy's acquisitions corps, many of those leaders expressed confusion and a lack of education regarding DevOps and its implementation within the U.S. Navy.

2.1 Fundamentals of DevOps

DevOps finds its genesis in the Lean Manufacturing movement of the 1990s [19]. The goal of Lean Manufacturing is to increase a business's profits by eliminating inefficiencies and reducing work in progress [20]. Lean also emphasizes real-time communication between team members as well as to those organizations outside of the development team [21]. These principles are predicated upon the need for complete transparency between all personnel about the state of the system and their work. This creates a tightly knit team of developers who can move rapidly and efficiently to generate and implement design changes.

In the early 2000s, the Lean Manufacturing methods were applied to software design and development in order to create faster, more flexible, and more valuable product development streams [22]. This new approach to software development is called Agile development. Agile seeks to find the same efficiencies as Lean Manufacturing by breaking the work of development into smaller, more frequent work periods called sprints [23]. These sprints are planned in Scrum meetings, which arrange for completion of a single feature or component of the software to be developed over the course of a few days or a few weeks, depending upon project complexity or difficulty [24].

The DevOps model takes Lean and Agile concepts one step further by seeking to integrate the development team, security and testing team, as well as those operating the system into

a single much larger socio-technical system. This socio-technical system is the summation of all of the human beings involved in the development and acquisitions process and the technology they use to accomplish their job [25]. With the developers, testers, certifiers, and operators all connected, the final step in a DevOps construct can be implemented, continuous feedback. In short, DevOps allows companies to leverage the innate intelligence of humans, their abilities to work efficiently in teams, and advanced technologies and automation to deliver better systems more quickly, more reliably, and with far more efficiency (e.g., remaining within budget) than previously practicable.

2.1.1 DevOps Testing

Like the Navy, private industry must test its products before releasing them to their customer. But unlike the Navy, testing within a DevOps system is performed constantly and whenever needed. The prevailing theory behind testing in an Agile or DevOps development environment is to break the software early and often so that weak points and inefficiencies in the code can be discovered and fixed quickly [26]. This is all part of one of the chief tenets of DevOps, which is continuous experimentation and learning [19].

This continuous learning is accomplished using a continuous testing and integration model. Figure 2.2 shows a typical process path. Along this path a developed piece of code is tested for functionality, then how well it integrates with the platform, and finally how well the total system functions, before being readied for deployment to the customer (What is known as "production"). Note that the testing is mostly performed automatically, often during the night or non-business hours [24].

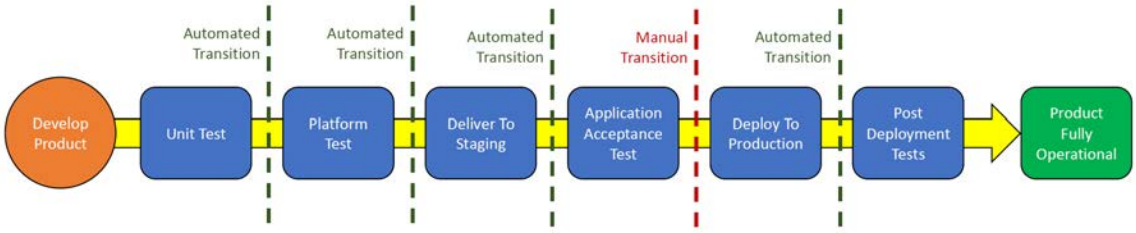

Figure 2.2. Nominal Continuous Integration Path. Adapted from [27].

This automated testing of bite-sized pieces of code with a much higher frequency than is found in traditional "waterfall" processes is a key element of any Agile or DevOps process that enables the rapid development and advancement of software in the private sector [3]. The testing equipment used is often special software programs that are designed to inject failure (In what is called fuzz testing [28]) into the operation of the system or attempt to penetrate layers of the system's security (In what is called penetration testing). This constant and continuous rate of to testing generates large amounts of data that is then incorporated back into the system's design to improve performance.

2.1.2 DevOps Environments

This testing relies upon the environment the system is tested in to be as close to the real world operating conditions as possible [29]. This recreation of real world conditions also extends into the development process, where great pains are taken to build equipment and software configurations in development and testing that directly mirror the production environment. This allows for the easy migration of any system from a development sandbox into a testing environment and then finally into production.

According to Kim et al. [19] this is the concept of continuous flow. This flow performs two functions: First, it helps developers to better understand the interactions that system they are creating will have with other hardware and software. This allows for continuous integration into an operating environment. Second, the this flow allows development teams to share and demonstrate their progress with management and customers. This makes the entire development process more transparent and allows management and customers to better grasp the technical risks and constraints of the development program.

2.1.3 Leveraging Big Data

This continuous experimentation, integration, and flow is designed to generate massive amounts of data. This data is collected by using automated monitoring and instrumentation of the system [30]. This data is then fed back to developers to better understand how it is being used and what the customer's needs are [31]. This data also helps developers catch bugs and security problems that were not found in testing prior to release. By better understanding how the customer uses the system and finding bugs as soon as they occur, developers are able to deliver a higher quality product.

But this sharing of information is not just confined to system data, it also includes sharing information regarding business practices and what works and what doesn't between development teams and all echelons of the business enterprise [32]. Because DevOps, like Agile and Lean before it, is as much a cultural phenomenon as it is a set of practices, it requires that teams share their experiences with each other. This helps the organization itself evolve into what Gibson et al. [13] refer to as a "learning organization." A learning organization utilizes a culture of educational leadership to fuel and reward creativity [33]. By rewarding creativity and encouraging all personnel to continuously improve their practices and education, the organization is more easily able to incorporate new ideas into their products and remain on the cutting edge of technology.

2.2 Who Makes DevOps Happen?

To transform a business into a learning organization requires not only the adoption of new processes and technologies but also new methods for managing personnel. Like Lean Manufacturing, Six Sigma, and Agile before it, DevOps requires the integration of teams and that an emphasis be placed upon communication between those teams [34]. This is why DevOps is not just a process or a philosophy but an entire socio-technical system since nothing can be accomplished outside of the context of the people executing the development, testing, and operation of the products of that DevOps system.

Truly, the most important aspect of any DevOps system are the personnel running it [16]. The way they are hired and organized can have an outsized impact on the performance of the program and the team [34]. In private industry, most companies are vertically integrated so that their operators, developers, and testers are part of the same co-located and intertwined team. This prevents the creation of silos of knowledge that impede communications and productivity [35]. Contrast this practice with the U.S. Navy (USN) and DoD, which often organize personnel in strict hierarchies that are separated by long distances. The typical division of labor in a traditional DevOps system contains the following three types of personnel:

- Operators: This group encompasses the personnel who operate and maintain technological systems. In private industry these are typically the system administrators and users who interact with the software on a daily basis to either perform tasks or

maintain the cloud infrastructure.

- Developers: This consists of the teams of engineers and programmers who provide the engineering, fabrication, and development work needed to bring systems to life.
- Testers: These are the personnel who conduct functionality, security, and bug testing. Often times these are the same personnel as the developers but there is a rising trend to use different development and testing teams to find defects faster [36].

2.3 DevOps In A Naval Context

Because of the confusion surrounding DevOps within the Navy, a universal definition for DevOps must first be established. This must be done so as to provide the Navy with a "conceptual sandbox" [37] that it can a operate DevOps system within. Investigating the definitions of DevOps used by leading practitioners such as Red Hat, Google, and Atlassian shows how varied each organization's approach to DevOps is. This illustrates how each practitioner of DevOps tailors the idea to their particular needs. This makes DevOps a difficult concept to define due to its amorphous nature. These definitions are collected in Table 2.1. Also included is the Navy's systems command (SYSCOM) in charge of all acquisitions and development of IT systems, Naval Information Warfare Systems Command (NAVWAR), which has also attempted to define DevOps for the Navy.

Table 2.1. Expert Definitions of DevOps

Practitioner	Definition
Atlassian [38]	"DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably."
Red Hat [39]	"DevOps describes approaches to speeding up the processes by which an idea (like a new software feature, a request for enhancement, or a bug fix) goes from development to deployment in a production environment where it can provide value to the user."
Amazon [40]	"DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes."
Google [41]	"DevOps is an organizational and cultural movement that aims to increase software delivery velocity, improve service reliability, and build shared ownership among software stakeholders."
NetApp [42]	"DevOps is an evolving philosophy and framework that encourages faster, better application development and faster release of new or revised software features or products to customers."
The DevOps Handbook [19]	"DevOps is the outcome of applying the most trusted principles from the domain of physical manufacturing and leadership to the IT value stream."
NAVWAR [4]	"DevOps combines the cultural and technical philosophies of software development, quality assurance, and IT/InfoSec operations into a single 'system of systems' that is managed as a whole."

By distilling the definitions of DevOps in Table 2.1 into a word map, common themes and ideas regarding DevOps become apparent. This word map sizes words based upon how often they appear in the source document. The larger the word, the more often it appears. A word map of the different definitions of DevOps from industry and government experts is shown in Figure 2.3.

Figure 2.3. Word Map of Definitions of DevOps Derived From Various Experts.

Looking at the word map in Figure 2.3, certain universal concepts become clear. The word map shows that processes and development are key aspects of DevOps, but also a broader cultural and philosophical change in how the organization approaches development. It is also clear that this philosophy is designed to emphasize the rapid development and fielding of software. Also prominent is the mention of software due to the development of DevOps as a process for software development. As will be discussed in greater detail later in this

thesis, the Navy does not have the luxury of only worrying about software when developing combat systems. The Navy must also find a way to apply DevOps to hardware development. From the word map and literature at large, the following seven key concepts of DevOps can be determined:

- Acceleration of Development
- Assurance of Quality and Efficacy
- Automation of Processes
- Focus on Capability
- Development of High Trust
- Philosophy and Culture
- Reliance on a Set of Best Practices

But not all of the definitions of DevOps have all of these attributes. As can be seen in Table 2.2, each business has tailored their definition of DevOps to fit their needs and their own cultural philosophies. In real world practice, this has turned DevOps implementations into dynamic and evolving socio-technical systems [43] that flex to meet the needs of the business. This means that businesses will focus on the aspects of DevOps that they perceive as being helpful to making their business more profitable or saving them money.

Table 2.2. Attributes of Experts' Definitions of DevOps

Practitioner	Acceleration	Assurance	Automation	Capability Focus	Philosophy	Practices
Atlassian	X	X	X			X
Red Hat	X			X		X
Amazon	X			X	X	X
Google	X	X			X	X
NetApp	X	X		X	X	
The DevOps Handbook					X	
NAVWAR		X			X	

As can be seen in Table 2.2, each definition has been customized to meet the needs of its user. In this same vein, the Navy must also tailor their definition of DevOps to meet their needs. This new definition must capture the unique operating conditions and needs that the Navy has while ensuring that it meets the intent of modernizing the Navy's processes. It must also take into account the need for combat systems to actually be placed in the hands of Sailors in a timely manner. This need to have an end product is just as important and the development thereof because "if you don't provide a product, DevOps is worthless" [28]. This new definition is as follows:

> *DevOps*: The convergence of collaborative culture and cyclical processes that increase the speed of development and the delivery of capabilities for Sailors while still ensuring the efficacy and safety thereof.

This new definition meets the Navy's need to keep Sailors needs first in mind throughout all aspects of system development. It also addresses the statutory need to prove that each system is reliable, capable, and safe. Furthermore, it implies the need for extensive feedback and clear communication between all parties involved in DevOps implementation. Without these elements, any DevOps system will be unsuccessful.

2.4 A Quick Summary

The key takeaway from this chapter is that DevOps is not just a new way of doing things but a new way of thinking about the way that the Navy goes about the business of developing and delivering combat systems. This will require a tighter integration of the development, testing, and operations of these combat systems into a symbiotic web of constant improvement. This will be a drastic shift for the Navy from Congress down to the individual Sailor on the deck plates. This culture shift is designed to maximize the personal responsibility each participant takes in the process and the amount of value that is provided to the customer [44].

For the Navy, that value is capability and lethality placed in the hands of each Sailor at sea and ashore. This value is gained by leveraging the key concepts of DevOps including operational feedback, continuous flow, and continuous experimentation. A full list of key concepts is included in Table 2.3.

Table 2.3. Key DevOps Concepts

Concept
Open communication and close collaboration
Continuous experimentation
Continuous feedback
Continuous integration
Continuous flow

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
What Stands In The Way Of DevOps?

With DevOps defined in Chapter 2, the next step is to define the environment in which the Navy currently operates and seeks to implement DevOps within. This is necessary in order to understand the challenges that the Navy faces in implementing DevOps within its acquisitions system. These challenges are categorized into cultural, organizational, regulatory, process, and technical challenges.

Throughout the ten interviews and through all of the correspondence gathered from respondents, a surprising trend became clear. Despite the technical nature of DevOps, the respondents' largest concerns were with the cultural, organizational, process, and regulatory hurdles that stand in the way of the Navy adopting a DevOps acquisitions system. Drawing upon these interviews, it became apparent that these hurdles are all interconnected and will require a holistic approach to change. These hurdles include the entrenched organizational culture of the Navy that is reinforced by the hierarchical organization of the Navy's Acquisitions Corps and the processes put into place during the 1990s to buy Cold War-era technology. As one respondent noted, the Navy's acquisitions hierarchy hasn't changed since 2002 [45]. These are further compounded by the statutory restrictions placed upon the DoD by Congress under Title X of the U.S. Code. The respondents all said that these non-technical problems must be solved prior to any technical solution being found. Despite this, there are two technical problems that must be overcome. These are the issue of how to perform DevOps for hardware and also how to create not only an infrastructure but also a supporting culture for collecting and using feedback data from the the fleet to design and build better combat systems. In the proceeding pages, these challenges will be explored and compared to the best practices from private industry and literature.

3.1 Cultural Challenges

Although the Navy's leadership recognize the need for change, this does not mean that all personnel within the Navy do. This leads to resistance to change that must be surmounted for the DevOps intervention to be successful. As identified by Gibson et al. [13], there are

four key reasons for change to be resisted. They are the self-interest of the resister(s), the resister(s) misunderstanding the change, the resister(s) having a different assessment of the best course of action for change, and a low tolerance within the organization for change. Within the Navy, the second and fourth reasons are the most pressing. Figure 3.1 illustrates the motivations for Navy personnel to resist this transition to DevOps and points out the two predominant reasons for that resistance.

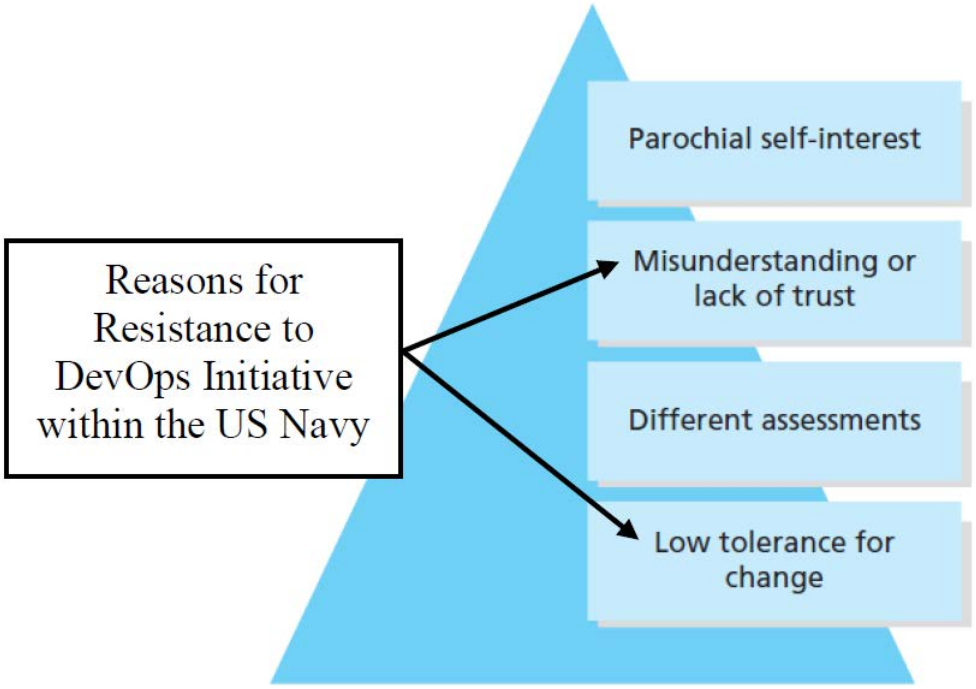

Figure 3.1. Navy Personnel Reasons for Resistance to Change to DevOps. Adapted from [13].

Since DevOps has only been a formalized concept since 2012 [19], there are many misconceptions about what it is. When combined with a history of missteps within the Navy's attempts over the last two decades (such as the "Paperless" Navy initiative [46]) that did not fulfill their promised goals, many business entities within the service are hesitant to trust a plan based upon what they perceive to be immature practices. Compounding this confusion is the Navy's historical dogmatic resistance to change due to simple institutional inertia [47] and a desire to preserve tradition [48]. As was noted by one of the respondents' who served

in the Navy, a Sailor's favorite phrase when asked why a task is completed a certain way is "That's the way we've always done it" [15]. This deep-rooted intolerance towards change when combined with mistrust of the Navy's leadership can make implementing change a nearly impossible task. But the Navy is not alone, the Air Force found that when they tried to implement DevOps practices within the F-22 Raptor Program that the hardest hurdle to overcome was the change in culture among its personnel [49].

One cyber security engineer stated that the misunderstandings about the nature of DevOps and resistance to change within the organization have hindered prior attempts to adopt DevOps or Agile business practices [45]. Once again, this is is reflected across the entire DoD. In an annual survey of major acquisitions programs in 2019, the U.S. Government Accountability Office found that of the twenty-two programs that claimed to be Agile, only six conformed to the best practices of private industry [2]. Compounding issues like these, the Navy also has its own peculiar cultural challenges that must be overcome.

3.1.1 Risk Aversion

Chief amongst these cultural hurdles is the the Navy's aversion to risk in acquisitions programs. As combat systems costs have increased and the Navy's budget has decreased since the end of the Cold War, the Navy's leadership has grown increasingly wary of any risks in their acquisitions programs [16]. This has resulted in a climate where, as one program manager at Naval Air Systems Command (NAVAIR) lamented, there is "no tolerance for risk within the Navy acquisitions and development hierarchy" [50]. This makes any deviation from established norms difficult to implement and stifles creativity. Part of the DevOps culture is creating an environment where personnel feel it is "safe to fail" [17]. The prevailing idea behind this is that failure is seen as a means to the final solution for a given problem and if the developers working on the project are allowed the leeway to take risks, they will be able to fail faster and solve problems quicker. This is also allows developers to be more creative and more flexible to respond to emergent system requirements.

This aversion to risk also arises in the regulatory environment in which the Navy operates. Of the ten respondents, eight lamented that the Navy's attempts to innovate are stifled by the rigid statutory regulations that are required by Congress to ensure that the government shoulders as little risk in acquiring new combat systems as possible. These statutory regu-

lations are then translated into DoD and Navy regulatory policies that dictate how to write contracts, how to decided upon contract awards, and how the government's money can be spent. As one consultant at Naval Sea Systems Command (NAVSEA) opined, "the reason we have the rules we have is because we messed up in the past and needed to codify rules to prevent those screw ups in the future" [51].

While these regulations reduce risk for the Navy and the government as a whole, they also reduce flexibility in how the Navy awards contracts and acquires combat systems. During the interview process for this thesis, every respondent expressed disfavor with the way that contracts were written and awarded. One acquisitions professional at NAVSEA stated that "we make contract awards based upon price alone" [15]. The manager for a program in the Air Force mentioned that "military contracts make things too specific" and because of that "we can't provide the best solution" to the warfighter [52].

This is directly opposed to the way that decisions are made in a DevOps system wherein decisions are made based upon ultimate value to the customer [19]. Managers work closely with customers to determine what the ultimate needs of the customer are and then translate that into sprint stories during the Scrum process [53]. These sprint stories dictate loosely what features the development team needs to design and implement over the next days, weeks, or months (Sprint duration is based upon size and complexity of features being implemented) to meet the customer's needs [17]. During these sprints, managers and customer advocates meet daily to ensure that the work being performed still aligns with the needs of the customer and the product will fit the desires of the customer regarding functionality or user experience. This close working relationship builds trust between the customer and the developer and nullifies the need for rigid contract language to ensure that the developer will deliver an acceptable product. In order to adopt DevOps, the Navy will need to overcome these trust barriers and build closer working relationships with its contractors.

3.1.2 Locked In With Vendors

The aversion to risk does not just affect the Navy, it also affects the contractors and vendors who design and build the Navy's systems. Because the development expenses are so high and the profit margins so thin, the contractors meticulously protect their intellectual property [16]. As one contractor mentioned, this results in the Navy relying upon proprietary systems

that require continued work from their creators to integrate with other systems or update them with new capabilities [54]. This forces the Navy to continue relying on single contractors to provide services or solve technical problems. This slows the process of development down and leads to political infighting between contractors around the awarding of contracts.

Contrast this with private industry that uses open source tools and software [3]. By relying upon open source solutions, private industry is able to leverage a larger pool of vendors and contractors, and therefore more possible ideas for solutions. As one consultant working on unmanned aerial systems remarked "We need to be better about designing open systems" so that "we're not tied to proprietary software or hardware" [54]. Failure to do so will cause continued problems within systems development programs. For instance, *USS Zumwalt* (DDG 1000) was initially designed to use computer servers from Microsoft, but during the middle of development for *Zumwalt*, Microsoft sold off its server hardware division. Because the requirements for the ship were written specifically for Microsoft hardware, this change led to unnecessary delays due to the slow requirements generation and approval process (Discussed in depth in Section 3.3.1). If the Navy continues to rely upon proprietary software and not open standards, it will continue to be unable to keep up with the blinding pace of change in both private sector and its adversaries.

3.1.3 High Reliability Organization

The Navy's risk aversion also extends into the operational realm. Because the Navy "has to live in a world where we kill people and break things" [55], there is little room for bugs or defects in the systems that are given to the fleet. This organizational mindset is one of a high reliability organization (HRO). An HRO is an organization that operates specialized systems that are deeply interconnected and that are hazardous and/or have a high risk of catastrophic error [56]. These interactions are complex and tightly coupled, meaning that errors are difficult to diagnose and propagate quickly throughout the system of systems or organization [57]. We see examples of these HROs in nuclear power plants and airlines.

Like other HROs, the Navy exhibits the following traits: "prioritization of both safety and performance and shared goals across the organization, a learning organization that uses 'trail-and-error' learning to change to the better following accidents, and a strategy of redundancy beyond technology but in behaviors such as one person stepping in when a task

needs completion" [58]. This means that the Navy must hold the systems and products it develops to a higher standard of quality assurance than other DevOps leaders like Microsoft or Google. If Google or Microsoft push out an update that breaks their users' systems, they simply have to launch a media campaign to apologize. If the Navy accepts a defective combat system, then Sailors and civilians will die. In the context of DevOps, this culture of high reliability results in approvals for release being slower to achieve, testing being more thorough, and requirements for quality control being more stringent. This will inevitably mean that the Navy cannot achieve the same level of development speed as the technology industry leaders it seeks to emulate since it has higher standards to meet. But, as one acquisitions professional at NAVAIR expressed, "DevOps may not make us too much faster but it's going to make us on time" [54].

3.1.4 No Champion of Change

The transition to DevOps will be a revolutionary change for the Navy. As Schein states, any change must have a leader who can model the change and motivate those in the organization to go through with it [59]. This change leader is responsible for communicating the necessity for change and carrying out any messaging plan regarding that change [60].

However, this communication cannot be just through verbal, print, or video messaging. It must also be performed through action. Simply put, leaders must "walk the talk" [60]. This creates a bottom-up generation of participation instead of what followers may perceive as simply direction from the upper echelons of the Navy. Leaders must also showcase themselves as exemplars of the DevOps culture. This means that they must interact with their followers and employees in order to generate engagement through the power of their relationships with their subordinates [33]. These leaders must also be open to feedback from their followers, in order to elicit their participation in the change process. The more that followers feel that their voice is heard, the more likely that they will welcome change. As one program manager mentioned, the change leader must "defeat the antibodies to change within the Navy's bureaucracy" [50].

This will be a major hurdle for the Navy. As one software engineer mentioned, the novelty of the idea of DevOps and the confusion surrounding just what exactly it is, has resulted in many in the Navy's upper echelons of leadership not understanding what must be done to

bring about change or how to communicate its necessity [28]. An Agile consultant stated that this is further compounded by the short duration in which leaders remain in command (Typically two to three years) and their high turnover rate [54]. This makes it difficult to carry out long term change leadership, especially for something like the adoption of DevOps that will likely take a decade or more to be fully realized. As one consultant at NAVAIR lamented, the Navy "needs someone at the [Senior Executive Service] or Flag level to lead the charge." That same consultant also stated that the Navy "needs a character like Hyman Rickover with passion, drive, and horsepower to get the organization charged and aligned" to the future of DevOps [55].

3.2 Organizational Challenges

Currently, the Navy's acquisitions force is organized as shown in Figure 3.2. Within this structure are the PA in green who are government program management, both in uniform and out, who make sure that the combat system is delivered on time and on budget. The offices in blue in 3.2 are those government personnel who manage risk for its development programs [15]. These personnel are called the TA and Technical Warrant Holders. TA personnel are often experts in their field and have decades of experience (They are often retired Sailors who worked on the systems in the fleet) with Navy systems [61]. These experts work for various systems commands such as NAVAIR, NAVSEA, and NAVWAR.

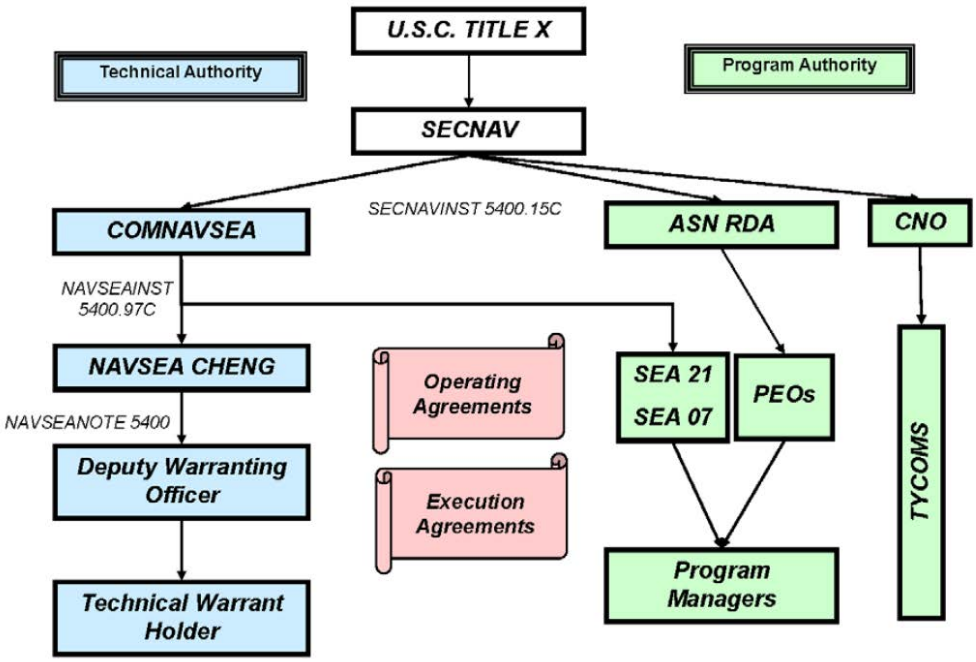

Figure 3.2. Current NAVSEA Organization. Source: [14].

The TA are aligned under the engineering divisions in each systems command as shown in Figure 3.3 for the engineering division for NAVSEA 05. They are responsible for certifying and managing the life cycle of every single piece of equipment that is installed aboard a ship or submarine [62]. This covers everything from solid state radars to nuts, bolts, and other fasteners. They are involved in the planning of testing by reviewing procedures to ensure that the tests effectively measure and demonstrate system capability in accordance with the system requirements. The TA then approve these tests to be conducted. Once the testing is completed, they review the test results and make recommendations for system certification [51]. Nothing can be installed aboard ship without the TA recommending its approval and the commander of U.S. Fleet Forces signing off on the final approval.

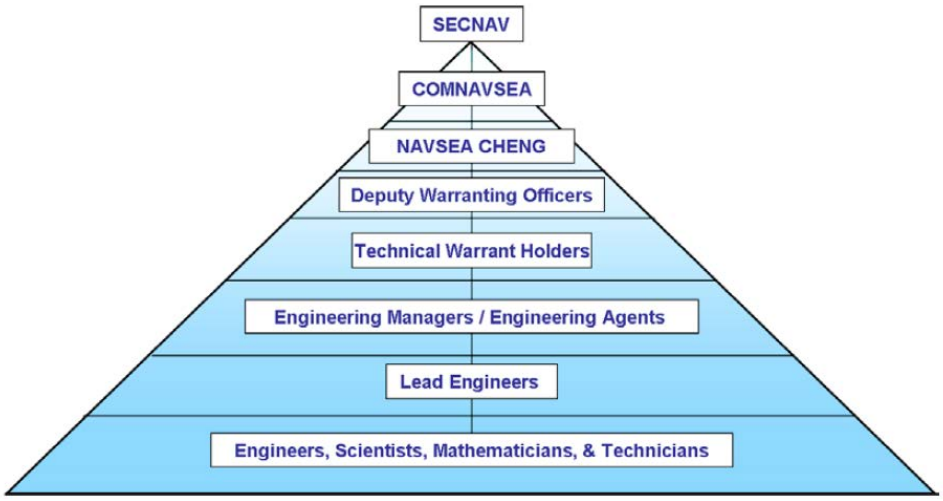

Figure 3.3. Navy Technical Authority Personnel. Source: [14].

The division of TA and PA personnel is not just administrative, but also cultural. As one program manager at Progam Executive Office (PEO) Integrated Weapons Systems (IWS) (PEO IWS) explained, whereas PA view their duty as getting new systems and capabilities to the fleet as quickly as possible, the TA see their job as preventing poor systems from entering the fleet [11]. This is due to the nature of the TA's job and who is selected to fill it. Due to their experience and duty to ensure that the combat systems being sent to the fleet integrate well with the rest of the Navy's equipment, they rely heavily upon statute and regulation to perform their duties. As a consultant for NAVSEA revealed, this creates a clash of cultures within acquisitions programs that slows progress and causes the job of the TA to be "largely theoretical instead of being meaningful" [15]. This conflict leads to a culture where decisions are pushed up the chain of command to the highest level of decision makers and, as one acquisitions professional at NAVSEA stated, "From the TA's perspective, procedural compliance is more important than delivering capability to the warfighter" [51]. This sentiment was echoed by another project manager at NAVSEA who said, "There are people who get lost in regulation and forget that their job is to deliver capabilities to the Sailor at sea" [61].

DevOps, on the other hand, requires that managers actively push decision-making to the lowest levels of authority within their organization. This is not typically done in today's Navy as the risk averse nature and low acceptance for loss of life within modern society

makes it difficult for commanders to allow their subordinates to make risk decisions that might end their career. But this reluctance is in direct opposition to the need for leaders within a DevOps culture to empower their followers to make decisions and take action [60].

Empowering developers to choose workflow or testers to modify testing procedures or criteria as needed to produce the best combat system possible is paramount to the proper execution of DevOps processes. Commensurate with this lowering of decision authority is the need to empower operators to request changes and provide feedback [13] to the developers and testers so that their needs are met and they feel as though they are actively engaged in the process. This means that the Navy must become a learning organization that rewards creativity as discussed in Section 2.1.3. To make this transition, leaders must foster a climate of high trust between all teams and groups within the Navy's Acquisitions Corps, which is based upon habitual open and honest communication.

3.3 Regulatory and Process Challenges

All the respondents were quick to point out that the Navy and the DoD are inundated with regulations and statutory requirements dictating how they will operate, acquire new combat systems, and perform development of new technologies. In technical terms this means that the Navy exists in what is known as an highly regulated environment (HRE). An HRE is an environment in which heightened security, access controls, segregation of duties, inability of personnel to discuss certain topics outside of specific areas, and the inability to take certain artifacts off premises are put in place [63]. An HRE is used when the intellectual property and methods being developed must be safeguarded from theft and all parties involved are sworn to secrecy. This directly conflicts with the need to share information openly and freely between all parties involved with the development of a system [19].

As an HRE the Navy has, as one contractor for an ACAT I program mentioned, "a certain level of stricture and structure that makes it harder to implement DevOps than the civilian sector" [55]. As private industries have to protect intellectual property, the Navy must limit the open sharing of capabilities, limitations, and technical details about its combat systems. This is due to the fact that combat systems development involves state secrets and the risk of them being leaked is a matter of life and death. This forces the Navy to have to work around "security concerns, classified information, non-ideal hardware restrictions," as well

as compartmentalization of vendors [55].

But the requirements for secrecy aren't the only regulations that the Navy must abide by when developing combat systems. The Federal Government also imposes strict requirements on the funding (10 U.S.C. §2433a), acquisition strategy (10 U.S. Code §2431a), and testing and evaluation (10 U.S.C. §2399 and 10 U.S.C. §139b/d) of new systems. This means that any new system must meet certain milestones and performance criteria prior to being accepted and that failure to do so may end in the program being canceled [16]. This contradicts the best practices of DevOps that dictate that the capabilities of a system should be built gradually. To place this in colloquial terms, DevOps requires that an elephant be eaten a bite at a time with small, frequent updates [27] whereas the DAS process requires the elephant be eaten all at once. DevOps would likely cause a program to be canceled under traditional criteria because it mandates that a system be placed into operation sooner rather than later, and that capabilities then be added on over time.

3.3.1 Inaccurate Requirements

This legalistic approach to systems development relies upon stringent documentation of requirements for the design of each system as well as the reasoning behind those requirements. This process is known as Joint Capabilities Integration and Development System (JCIDS) and is managed by the Chairman of the Joint Chiefs of Staff via the Joint Requirements Oversight Council (JROC) [64]. This JCIDS process is the initiation of any acquisitions program under the DAS and forms the basis for all design and engineering decisions that will be made within that program. It is illustrated in Figure 3.4.

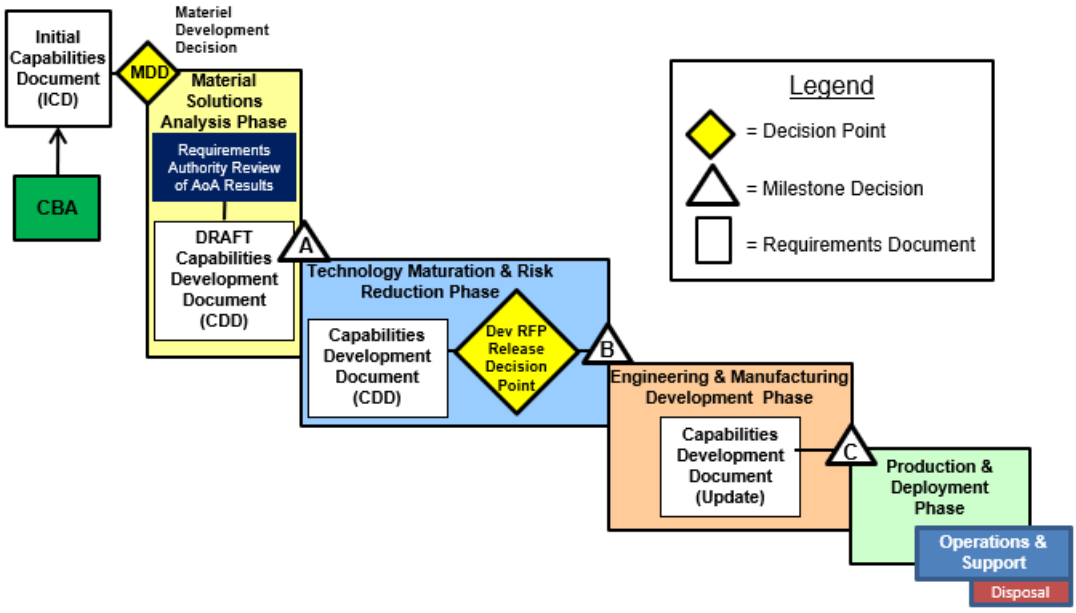

Figure 3.4. Joint Capabilities Integration and Development System Process. Source: [65].

The JCIDS process begins with a capabilities based assessment (CBA) that compares the Navy's request for combat systems against the list of military capabilities maintained by the JROC [65]. This list of military capabilities contains all of the functions that the entire United States military should be able to accomplish. This is built based upon input from each service, which for the Navy is Office of the Chief of Naval Operations (OPNAV) N8 [64]. As one contractor at NAVSEA explained, this list of capabilities is used to determine the requirements that any combat system should be able to meet that are incorporated into the Initial Capabilities Document (ICD) [15]. This ICD then informs a materials solution analysis that determines whether or not the Navy should move forward with developing and procuring new systems. Thed aforementioned contractor further explained, this decision is milestone A in Figure 3.4 and rests upon the shoulders of OPNAV N9, who then allocates funding for the program [15]. After funding is allocated, a CDD is developed that describes exactly how the Navy will meet the capabilities that they have identified in the ICD [66]. This CDD must be approved by the JROC (Milestone B in Figure 3.4), but upon approval is used to define and justify all funding allocations and staffing for a development program. All decisions in design, engineering, and testing must be justified with reasoning from the

CDD.

On paper, these documents should incorporate feedback and input from Sailors and officers in the fleet. Unfortunately, that isn't the case as an acquisitions consultant at PEO IWS relayed, "Currently Sailors have little say in what goes into combat systems" [15]. Furthermore, the requirements written in the CDD are often written to describe specific functions instead of outcomes for the fleet [28]. For example, the requirements derived from the CDD are often written to describe specific tasks instead of outcomes that reflect "what the Navy needs" [28].

This is in opposition to how design requirements are defined in an Agile development environment or DevOps system that put the customer's needs as the top priority [23]. These needs are captured during daily or weekly scrum meetings in which the sprints (Development periods) are planned. In each of these scrum meetings, there is a customer advocate who champions the needs of the customer to ensure that the finished products are satisfactory [19]. The guiding principle in private industry is to provide value to the customer and focus on outcomes for them [22]. This means that if the finished product is functional but doesn't provide exactly what the customer is looking for or the user experience is sub-par, then it is considered a failure. In this scenario, the possibility of failure is avoided by meeting frequently with the customer advocate to review the progress being made and determine whether what is being developed still meets their needs or not. By implementing direct feedback from the customer, the developers are better able to provide successful products.

3.3.2 Cumbersome Certification and Testing Process

The Navy is not a unique entity in their requirement to test and certify their end product. In fact, testing is an integral part of the DevOps process. But unlike private industry, where all testing is rolled into one constant cycle, the Navy conducts testing in finite, linear stages that are tied directly to milestones in the combat system's life cycle (Reference the milestones in Figure 1.1). Furthermore, the Navy differentiates between two different types of testing and evaluation. These are developmental test and evaluation (DT&E) and operational test and evaluation (OT&E) [67]. DT&E is performed during the technology development and engineering and manufacturing phases of development (As shown in

Figure 1.1). This testing is designed to prove design concepts, demonstrate technological maturity, and identify integration problems prior to final prototyping. OT&E is carried out at the end of the engineering and manufacturing phase utilizing the final prototype systems. The objective of OT&E is to determine if the system is operationally effective (i.e. can it accomplish its mission) and operationally suitable [61].

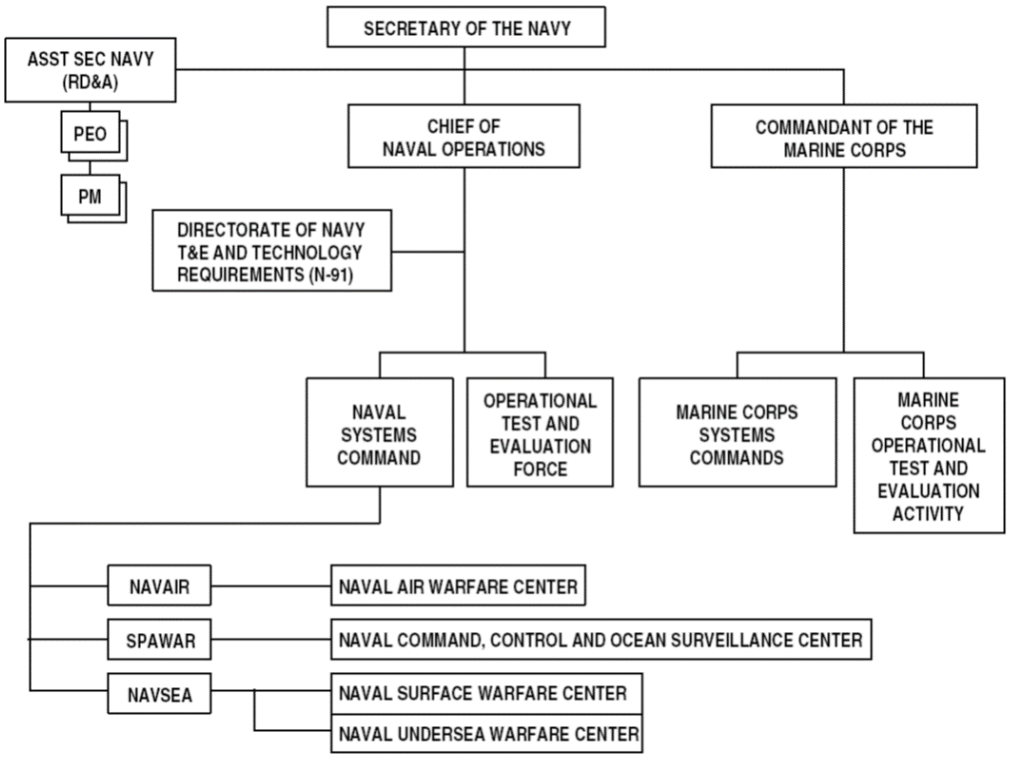

Figure 3.5. Navy Testing Command Hierarchy. Source: [67].

As shown in Figure 3.5, testing within the Navy is carried out under the authority of the PA with either the contractor (In the case of DT&E) or Commander, Operational Test and Evaluation Force (COTF) (In the case of OT&E) performing the testing. This testing must be conducted within the guidelines of the DoD's Director of Developmental Test and Evaluation (DDT&E) and Director of Operational Test and Evaluation (DOT&E), respectively [67]. All testing is based upon and conducted in accordance with each program's test and evaluation master plan (TEMP), which is derived from the capabilities documents produced during the initial design phase of development via the JCIDS process. This TEMP

ties testing events to specific capability requirements as well as to development milestones and serves as the "contract between PA, TA, the contractors, and COTF for what is to be tested" as one contractor at NAVSEA spelled out [15].

This testing procedure was born out of a need for the Navy to develop complex hardware systems and to prove their efficacy prior to delivering them to the fleet. This need to develop hardware alongside the software forces the developers to have to design tests and divert resources for the test equipment necessary to perform those tests for the hardware [68]. This means that any development of combat systems using DevOps must include detailed test planning [3] and plentiful developmental testing early in the project [68]. Hardware development also results in rigid testing schedules that do not respond well to changes or delays. As one expert in how the Navy performs testing revealed, the TEMP usually takes years to get approved as it has to be reviewed by PA, TA, the Navy N9 office (Who is responsible for system requirements and resource allocation), DDT&E, and DOT&E [51]. This expert's example was the TEMP for *USS Gerald R. Ford* (CVN 78). The *Ford's* TEMP took ten years to make it "through the labyrinth of bureaucratic red tape" for approval [51] because every time a change was made in the technology being used during the ship's decades of development, the TEMP had to be updated and go back through the entire approval process from the start. This delayed testing and ultimately the final delivery of the ship to the Navy.

Contrast this with the way testing is performed in a DevOps environment where the prevailing theory is to break the software early and often so that weak points and inefficiencies in the code can be discovered and fixed quickly [26]. Using these Agile testing practices, a system can be updated and improved rapidly due to the massive amount of data available to the developers to identify problems and adjust code or hardware components. Once again, the goal for testing in a DevOps environment is to shorten the time it takes to build a system, test it, and put the results from those tests back in the hands of developers [36]. Like the Navy, there is a need for strategic planning of testing to ensure that the testing is adequate for pushing the system to its limits and testing its functionality. Often times this planning is performed using software that integrates directly with the testing suite to provide better collaboration throughout different departments in the company [69]. Also like the Navy, private industry leverages cross-team testing where the team responsible for testing is different than the team that developed the product [36]. It is claimed that this cross-pollination

of testing and development teams allows for the detection of defects faster. The Navy must adapt its current testing practices to provide for better cross-team collaboration and a higher volume of tests in a shorter amount of time.

3.3.3 Sluggish Software Certification and Testing Process

Software systems have a second analogous process that they must undergo in order to be approved for use on Navy computer networks. This process is part of the DoD Information Assurance (IA) Certification and Accreditation Process (DIACAP) and results in the software earning an authority to operate (ATO) certification [16]. The main focus of the DIACAP is to ensure the security and integrity of the DoD's IT systems. Similar to the systems certification process, software must be tested against security requirements, those test results must then be reviewed by an authorizing official (AO), and upon successful completion that AO issues the ATO allowing the software to be loaded onto Navy computers and servers.

Like the systems certification process the DIACAP moves at a glacial pace and requires manual approvals and initiation of testing at specific program milestones [4]. Every expert interviewed mentioned that the DIACAP moves too slowly and the requirements needed to achieve an ATO are too cumbersome. A computer scientist at NAVWAR mentioned that the ideal IA process would allow for a continuous ATO as well as cross-compatibility between systems so that software is able to be loaded on any system once it is deemed "safe" [28].

Now contrast the glacial pace of the DIACAP with the continuous Risk Management Framework (RMF) process that is typically used in private businesses [70]. The RMF applies the same ideas of continuous learning and integration discussed in Section 2.1 to information security in order to reduce the time it takes to detect security threats and respond to them [12]. The RMF does this in a simple process of identifying potential risks, prioritizing those risks based upon their threat to the customer and developer, developing mitigation strategies, enacting those strategies, and then testing them [70]. That test data is then fed back into threat assessment. This means that the threat assessment is a continuous, ongoing process instead of a singular event.

3.4 Technical Challenges

Though the non-technical impediments took center stage during the interviews, there were still two technical solutions that the respondents were not sure how to solve. These challenges are how to perform hardware development using DevOps or Agile methods and how to implement data feedback loops while still fulfilling the requirements for data security and classification. The three engineers who were questioned were confident that technologies and software used in private industry could be adapted to the purposes of the Navy's acquisitions programs. As one mentioned, private industry "has been doing [DevOps and Agile] for years" and the software is "out there" and readily available [45].

3.4.1 Hardware DevOps Is Difficult

The largest technical challenge preventing the Navy from implementing DevOps is the fact that the combat systems the Navy needs are comprised of both hardware and software. Software is built in the aggregate, with each update and patch serving as a building block towards the overall capability of the system [68]. This iterative process results in software being augmented in small iterations and features being added until the code base or the computational resources can support it [71]. Inversely, in hardware programs this infinite addition of capabilities is what is known as features or requirements "creep." This creep often causes ambiguity in the primary capabilities of the system, difficulties in systems integration, sub-par systems performance, cost overruns, and eventually project cancellation [68]. This is why the DAS process is so rigid, so that performance requirements are defined early in the design process and that costly design revisions and physical rework during manufacturing can be avoided.

To date, there has been very little literature written regarding the adaptation of DevOps or Agile methodologies to hardware development. This is due to the nature of hardware development and the need to invest heavily in up front material costs for hardware as well as in testing equipment [72]. The usage of Agile or DevOps practices within hardware development is also confounded by the need to develop hardware upon which to run software [73]. Unlike pure software systems and development programs, many of the systems like radars and missiles are not hardware agnostic and need specific hardware developed to meet operational needs. This means that the hardware must be developed before or in

conjunction with the software. This forces the need to find ways in which to divorce the development and updating of software away from hardware [15].

This abstraction of the software away from the hardware would make for simpler development programs and allow software development to be unhindered by hardware limitations. As one scientist at NAVWAR explained, "The hardware update tempo is much slower. Software can be updated daily but hardware takes years" [28]. That being said, due to the Navy's culture as an HRO, this means that any integration of hardware and software must still be able to function safely and reliably. Furthermore, hardware becomes obsolete much faster than software. In the current "waterfall" DAS that takes a decade or more to come to fruition, this obsolescence of hardware creates a "tech refresh spiral" that leads to nearly "endless requirements creep and the eventual death of programs" [50]. These facts mandate that any adoption of Agile or DevOps methods in the hardware domain make sufficient use of configuration management tools to ensure functional integration of differing levels of hardware maturity. As the same scientist at NAVWAR clarified, "The goal isn't how to do Agile hardware but how to manage obsolescence" [28].

3.4.2 A Need For Data Feedback

As discussed in Section 2.1.3, any DevOps system is dependent on the ability to collect and distribute continuous feedback on combat systems back to the developers. While this is not an entirely foreign concept to the Navy, as test data is required for the DIACAP and systems certification process to function, it is by no means automated or continuous. As a testing manager at NAVSEA relayed, the Navy currently relies upon instrumentation for tests that must be installed manually and combat systems must be configured to collect and store detailed data [15]. That data must then be packaged manually and couriered back to developers and engineers for analysis as there is no automatic system to collect and transmit that data back ashore to developers [74].

But not only does the Navy lack the infrastructure to automatically collect and distribute the data that's been collected, it also lacks the personnel needed to make sense of all of the data. As one software engineer at NAVWAR explained, data scientists are often used to in private industry to analyze and interpret data to answer questions such as "Are we effective?" or "Can we accomplish the mission better?" [28]. These data scientists play a crucial role in

finding connections between the data and root causes of sub-par performance. They can also play a role in better understanding customers needs. For instance, when a customer says that user interface is "bad" the data scientist can perform analysis and find data to show that what the customer meant by "bad" was actually slow loading times.

This kind of interpretation for the customer is no less important in the DoD. As one assistant program manager at PEO Soldier explained, the Army needs the data feedback and analysis to understand "how better physical training scores correlates with better marksmanship" [75]. Unfortunately, the Navy and the DoD as a whole lack the number of data scientists needed to support all of their acquisitions programs [51]. This is a critical need that must be filled for DevOps to work.

3.5 A Quick Summary

The transition from the current "waterfall" DAS to a DevOps systems will face significant challenges. These challenges are summarized in Table 3.1 and include both statutory restrictions and regulatory policies. Perhaps the most daunting task will be changing the culture within the Navy's acquisitions corps and fostering more open communication between all parties. This will take diligent efforts from the Navy's leadership to lead the change that they wish to see in the Navy.

Table 3.1. Map of DevOps Concepts and Associated Challenges

Concept	Challenges
Open Communication and close collaboration	• Rigid organizational hierarchy • Secrecy requirements • Cultural inertia
Continuous experimentation	• Statutory requirements • Rigid test processes • Rigid JCIDS process
Continuous feedback	• Lack of infrastructure • Secrecy requirements • Cultural inertia
Continuous integration	• Cultural inertia • Rigid test processes • Rigid JCIDS process • Hardware requirements
Operational flow	• Entrenched cultural practices • Rigid work processes • Rigid JCIDS process • Hardware requirements

CHAPTER 4:
What Does A Navy DevOps System Look Like?

Simply identifying the challenges facing the Navy's adoption of DevOps without providing solutions would leave this thesis impotent and not worth the paper it is printed upon. This chapter will propose solutions to overcome the hurdles facing the Navy and hopefully provide a new perspective to shift the outlook of the Navy's personnel and leadership. It will address the changes necessary in both software and hardware development as well as personnel structure.

Before any of these solutions can be proposed, role of personnel within the entire DevOps system must be fully understood. Like in private industry, there must be those do the engineering and design work as developers and then there must be those who operate and maintain the combat systems as operators. Unlike private industry, federal regulations require that there be outside entities such as TA and COTF who test and certify combat systems before final deployment to the fleet. Using the same paradigm as Section 2.2, the roles of Navy personnel within this new DevOps system become apparent. These roles are recorded in the following list:

- Operators: This group encompasses the civilian and uniformed personnel who operate and maintain the Navy's combat systems. Aboard ships and submarines, these are often times separate personnel who have different concerns. The Sailors who operate the systems need them to be intuitive and capable at a moment's notice whereas the Sailors maintaining the systems require them to be easily accessible and modular for high maintainability. Ashore, the personnel maintaining and operating combat systems usually consists of the same civilian and uniformed personnel.
- Developers: This consists of the civilian contractors who work for large corporations such as Lockheed Martin and Raytheon who provide the engineering, fabrication, and development work needed to bring combat systems to life. It also consists of the government program management personnel, both in uniform and out, who make sure that the combat system is delivered on time and on budget. These government personnel are often referred to as the PA.
- Testers: These are the Navy's personnel dedicated to demonstrating that the system

being developed is capable of doing what it is designed to do. These testers are drawn from the DoD DOT&E staff as well as the Navy's COTF [67]. These personnel design and execute testing based upon the system's CDD and the input from the Navy's systems experts. The testers are the lynch pin in the Navy's acquisitions process. Without their ability to conduct testing, then the Navy is not legally allowed to accept the delivery of a new system.

With the roles and responsibilities clearly delineated, they can be applied to the DevOps framework illustrated in Figure 2.1. By inserting a certification element into the cycle, and illustrating where the brunt of the effort of both the developers and operators exists, the cycle can be adapted to the Navy's uses. This is shown in Figure 4.1.

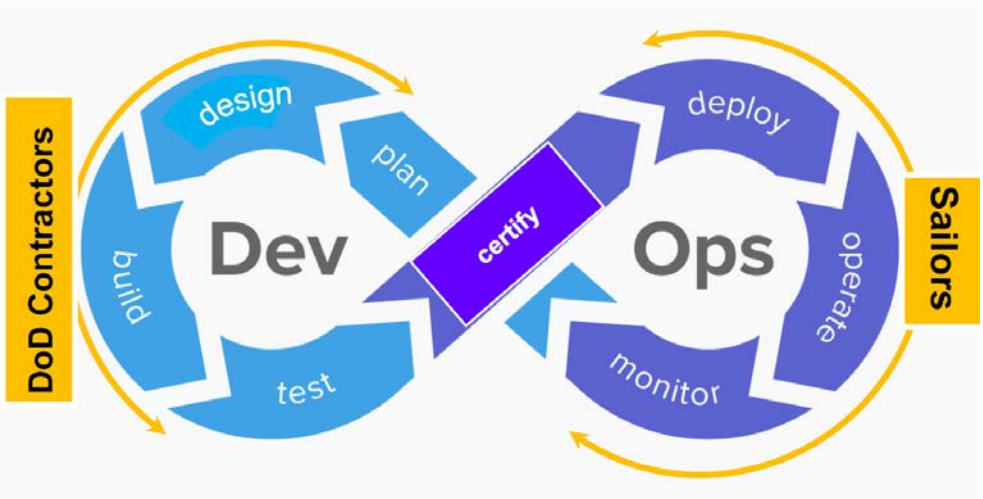

Figure 4.1. Proposed DevOps Cycle. Adapted from [18].

As is clearly depicted, the contractors are responsible and bear the majority of the burden of work during the design and build phases of the cycle since this most closely aligns with the Technology Maturation and Engineering and Manufacturing Development phases of the traditional DAS process prior to milestone C (as explained in Section 3.3.1). Inversely, Sailors and Navy civilians are responsible for the majority of the work during the deployment, operation, and monitoring phases as these find their analogues in the Production and Deployment as well as the Operations and Support phases of the DAS process. During the test and certify phases of the TA and COTF personnel would step in to perform independent

testing of combat systems to determine whether the systems can meet the requirements set for them safely and effectively. Also, during the planning phase of the cycle, all personnel would meet to discuss the needs of the Sailors at sea, develop system requirements, and develop a TEMP to demonstrate those requirements. This new cycle would allow for a simpler dissection of labor into short sprints and

4.1 A Flexible Organizational Structure

This new DevOps system cannot be implemented within the Navy's current organization structure. As discussed in Section 3.2, the Navy's current organizational structure is too rigid and unyielding to foster innovation necessary to keep the Navy competitive in the 21st century. Of the ten respondents interviewed, seven noted that the Navy's current organization was inadequate to build the information and communication pathways necessary to quickly deliver new or updated capabilities to Sailors. One cyber engineer stated, "We need to find a way to sit developers next to Sailors" to better communicate their needs and "get our Sailors exactly what they need" [45]. Bringing the technical warrant holders, program management team, DoD contractors, and operators more closely together will blur the line between each group, but will foster a collaborative environment within which the entire acquisitions team can implement new ideas faster and more efficiently. This will still need to fit within statutory restrictions but must still flatten. That flattening is illustrated in Figure 4.2.

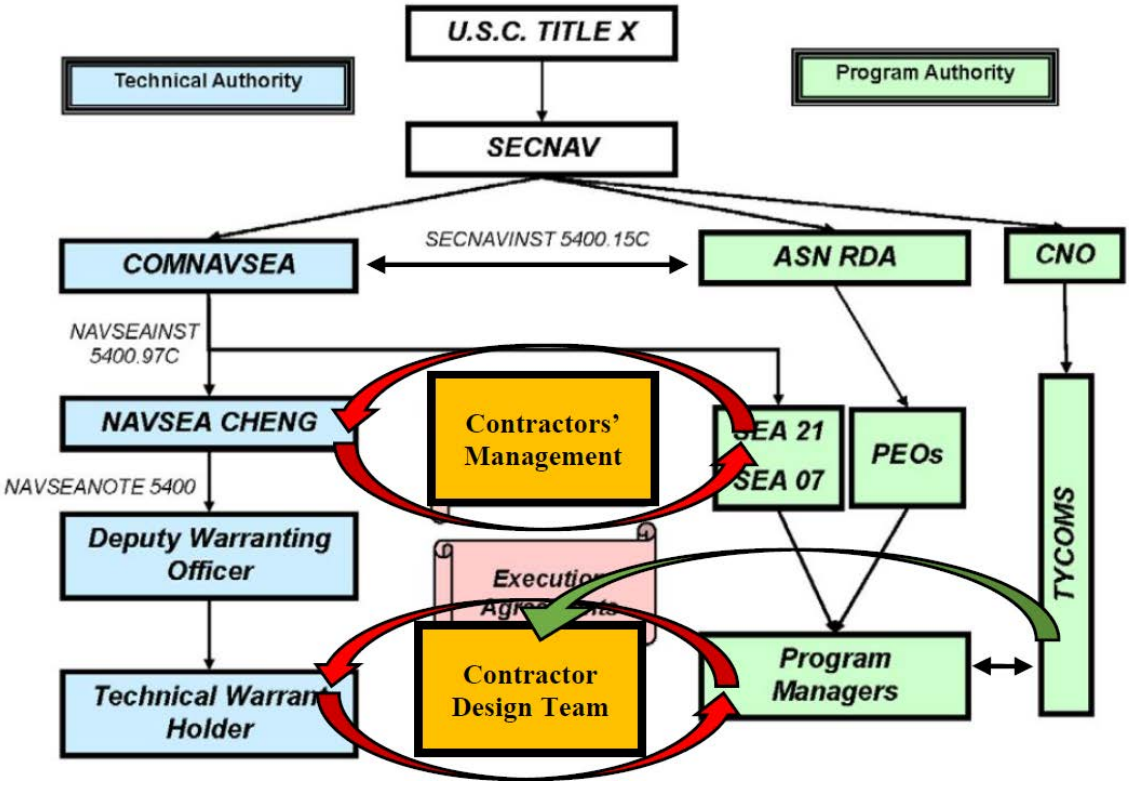

Figure 4.2. Initial Proposed Modified NAVSEA Organization. Adapted from [14].

This flattening of hierarchies would create liaisons and a better information pathway between the technical warrants, DoD contractors, and program management team (a shown with red arrows). This would allow the free exchange of ideas and more collaborative problem solving. This collaboration must occur at both the engineering level as well as the management level between the officers of the DoD contracting companies, the Chief Engineer at NAVSEA, and PEOs in charge of each family of combat systems. This will ensure that policies and organizational decisions will align with the efforts being made with the development of the combat systems themselves.

Of note is the increased interaction between the Assistant Secretary of the Navy for Research, Development, and Acquisitions (ASN RDA) and the Commander of NAVSEA that is needed. This would allow the alignment of Navy-wide policies and regulations to better

mirror the work being performed at the engineering level. This represents a shift from a top-down to a bottom-up decision-making process. Also, this would better integrate feedback from operational commands (denoted as Type Commanders (TYCOMs)) into the development process so that their needs are better captured during systems design, integration, and testing. This would be facilitated by the program management team as denoted by the dual-pointed arrow between them and the TYCOMs.

But this flattening of the hierarchy must still be translated into a real organizational structure. One answer is found in NAVSEA's own Engineering Technical Authority Manual. According to the Manual, TA personnel are often incorporated into an Integrated Product Team (IPT) to assist developers with the task of integrating their particular system into the rest of the ship [14]. This brings the TA and PA personnel together and integrates the decades of experience and systems expertise of the TA personnel into the design of the system. In theory, this ensures that the new system will be able to integrate easily with the current hardware and software installed on the ship as well as fit into the space available. While in practice this may not be the case (None of the respondents from NAVSEA had seen one of these IPTs performed), it still provides a useful framework to build upon. The Air Force has already implemented a similar construct. When interview, the program lead for an Air Force software development program stated that they had already organized their program office in a manner similar to this [52]. Their office structure combines contractors and uniformed Air Force personnel in the same office both performing coding and day-to-day development work. They also incorporate their own security testing personnel into the same office. This allows an environment where developers, operators, and security testers can easily collaborate and discuss ideas with each other without having to rely upon teleconferencing or lengthy e-mail chains.

Translating this into the Navy's current hierarchy, a new structure could be constructed that easily fits within the current statutory requirements. By leveraging the concept of IPTs the Navy can bring together teams to work on specific combat systems capabilities or improvements. These teams would consist of developers from contractors, project managers from the respective program office, and customer advocates from the respective TYCOM. These IPTs would also include subject matter experts from the TA office and COTF personnel to assist with test planning and execution. This new organizational structure is depicted in Figure 4.3.

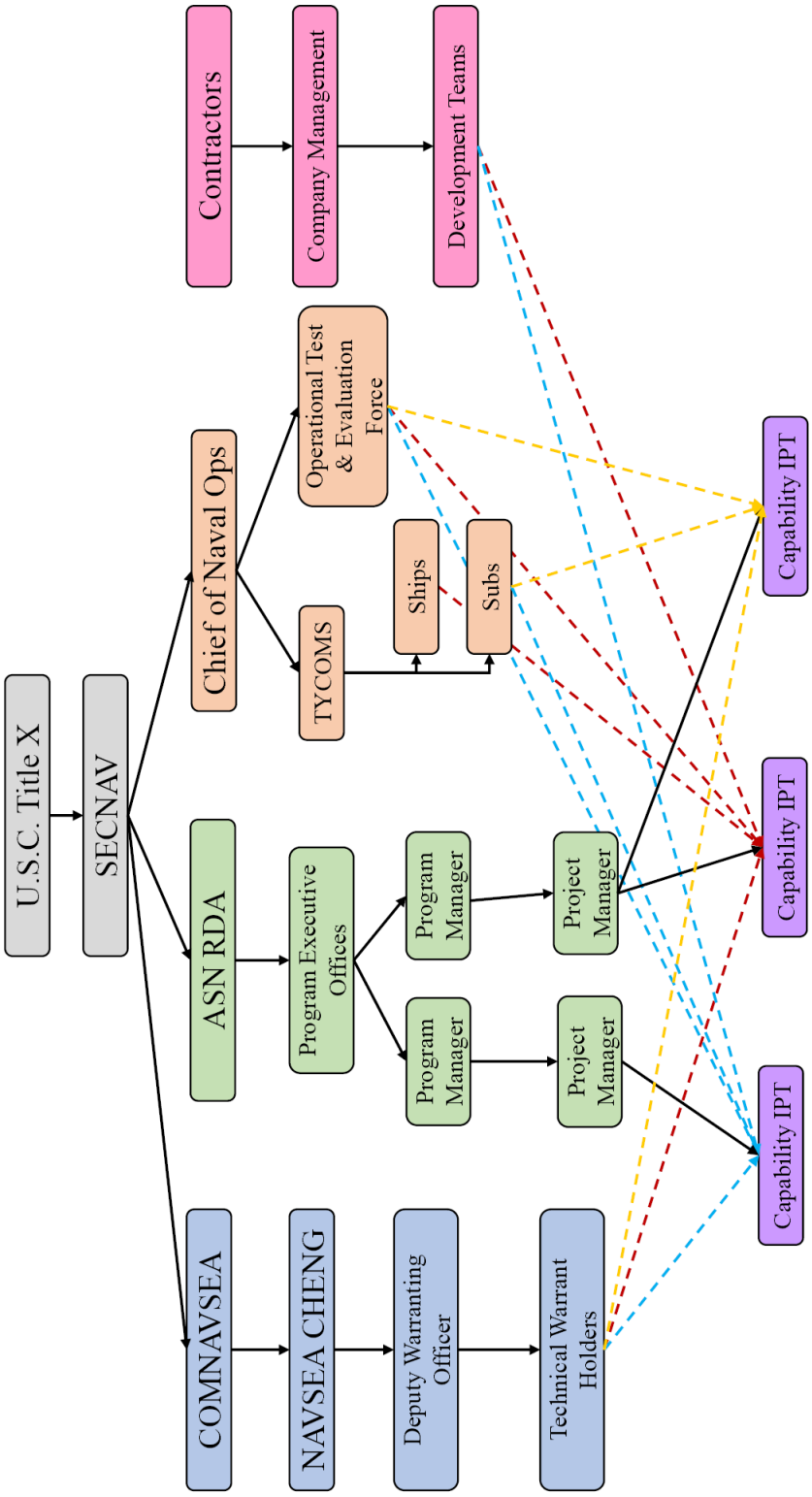

Figure 4.3. Proposed NAVSEA Organization for DevOps.

As can be seen, much of the same Navy hierarchy that is mandated by federal statute remains in place. This is a purposeful decision in order to ease the transition to DevOps gradually by incorporating its tenets into existing structures. Ideally, this would structure would change over time and morph into a final form, but that form cannot be predicted during these infantile stages of transition. But by relying upon concepts and organizational ideas already familiar within the Navy, more support for this change can be generated.

Work would be divided to each of these IPTs per capability or project they were working on. The program office or PEO would ultimately have authority over the IPTs for their program but outside personnel would be assigned to the IPTs directly. Work management would occur via Scrum meetings at varying intervals depending upon the span of control of each office. Using the example of the IWS 10 program office shown in Figure 4.4, the IPT for user experience upgrades would meet likely meet daily to plan and manage the execution of sprints. Each of the IPTs under the control of IWS 10 would then meet weekly with the IWS 10 program manager to plan and manage the work for the entire program. Then all of PEO IWS would likely meet monthly to manage work within the entire department. Of course these meeting intervals could be adjusted by each program as the PA staff sees fit.

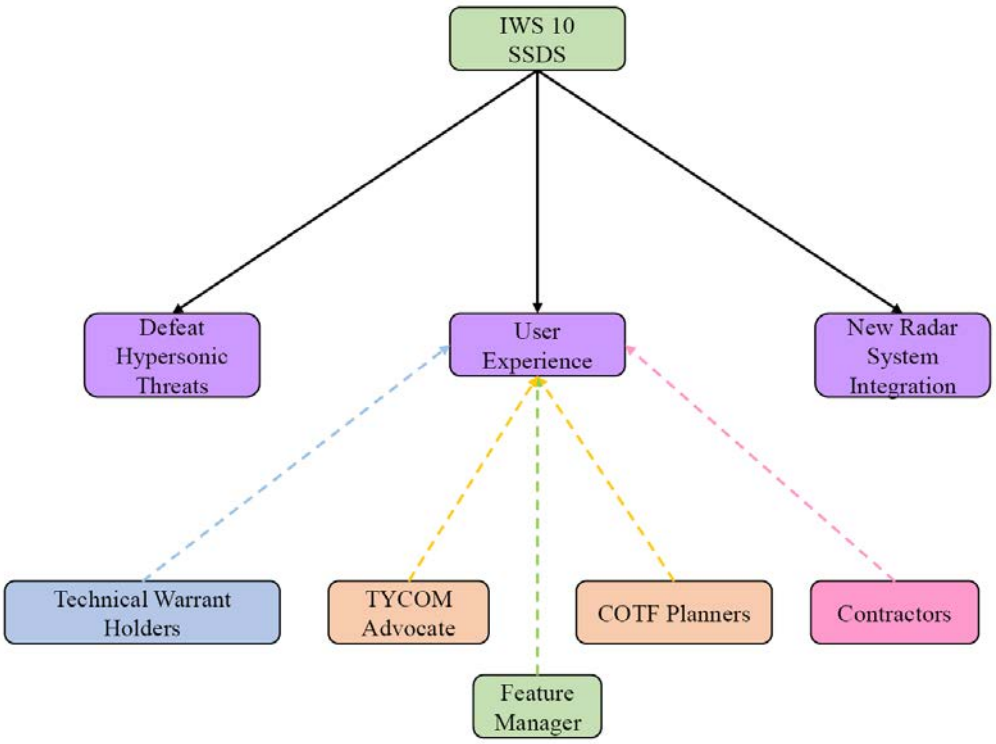

Figure 4.4. Proposed NAVSEA IPT Structure for DevOps.

This usage of granular IPTs also has other benefits. First, it provides for the siloing of information to only those who need to know. This will allow for administrative control over access to intellectual property and state secrets so that only those working on each projects can share that classified information. Using Figure 4.4 again, the user experience IPT would not have access to information concerning hypersonic weapons or radar performance data. If they do need clarification or assistance from those other IPTs, then they will meet with product experts who can answer their questions and assist them without revealing protected information. A contractor at NAVAIR already relies on a similar practice for the development of an unmanned aircraft system (UAS) for the Marine Corps [54]. When the team was presented with the need to integrate a classified component into the unmanned system, the development team for the classified component sent a liaison to NAVAIR to participate in Scrum meetings and assist engineers during design. As the contractor recounts, the team required "another customer advocate in the sprint and Scrum sessions to bridge

the gap and advocate for the needs of the classified subsystem" [54].

The second benefit of this IPT model is that it provides for the easier co-location of all personnel within the IPT. Unlike the current development environment of the Navy where the various personnel involved are heavily distributed across the country, each IPT can be distributed, with program office and SYSCOM management performed remotely. This will allow all personnel within an IPT to work in the same physical office and better collaborate and share information and ideas, which is necessary for a functioning DevOps system.

4.2 A New Approach to Development

To complement this new organizational structure, the Navy must also change its approach to the way it develops combat systems. This new method must adopt the best practices of Agile and DevOps systems. This must be done to build better cooperation between developers and operators, capture systems requirements more accurately, and build more open, flexible combat systems. The overarching goal of the DevOps system is to separate the updating and upgrading of software and hardware so that incremental changes can be integrated into systems that are already deployed exactly when needed to provide the lethality and survivability that Sailors require.

Part of this change in development practices is to a develop a requirements generation process that it is fast, flexible, and accurately captures the needs of Sailors. As discussed in Section 3.3.1, the current JCIDS process does not meet these needs. Part of the reason for this is how rigid the requirements process is under current JCIDS and DAS regulations. As explained by a senior scientist at NAVWAR, "In AGILE you have to be able to trade off requirements, cost, schedule, but in waterfall you can only trade off cost and schedule. Even if you are chasing unrealistic requirements" [28]. The other reason is that modern combat systems have become too complex to accurately capture designs and requirements via paper documents. A senior contractor at NAVAIR explained further, "You have to have your hardware and software design linked and you have to understand your interfaces. The complexity of our systems is so great now that we are unable to do that on paper and have to move to electronic means" [54].

4.2.1 Iterative Requirements Generation

To solve the problem of inaccurate and inflexible requirements will require a fundamental change in the way that requirements are generated. To accomplish this, requirements will need to be written in "looser" verbiage and also come directly from needs identified by Sailors and other operators. This new looser verbiage must be specific to what is being developed but cannot be too specific. As one Scrum Master with the Air Force explained, "DevOps needs freedom to operate and be creative and military contracts make things too specific. You can't do Cost-Schedule-Performance in the DevOps process or do hard requirements because halfway through you might need to pivot" [52]. Also, these requirements must come directly from the people operating the combat system and relying upon it in operational settings.

Under the new DevOps organizational hierarchy, each IPT would rely upon the TYCOM personnel embedded within them to provide insight as to operational needs and employment of the system being developed and then they and the PA personnel would capture those as system requirements. These inputs would then be used to plan sprints (The development period within Agile Methods) and generate a project or program "roadmap" regarding the schedule of development. As a project manager at the BESPIN Program mentioned, the "roadmap is crucial for your customer and team. It is very vital to know where the product is going and have vision for the end product. It also helps with contracting and funding allocation to show rough timelines" for product development [52]. This "roadmap" would replace the traditional "waterfall" or Gantt charts traditionally used to plan current development programs. The "roadmap" would then be reviewed at the end of each sprint for accuracy, delays, and re-prioritization of development goals.

But the "roadmap" and TYCOM input are not enough. The way requirements are captured must also change. Instead of being strict orders for what a system must do, they must instead be what one engineer working for a NAVAIR program called "statements of work built broadly" to encapsulate the full spectrum of use for the system [55]. For the Marine Corps' Marine Air-Ground Task Force (MAGTF) UAS Expeditionary (MUX) Program, as one consultant explained, that "at end of sprint, we do a sprint review with active Marines and former Marines as customer advocates and have those Marines deconstruct concept of operations diagrams to understand what the UAS needs to do from an operational standpoint" and that they then use this feedback and "concept of operations documents and

requirements letters from the Marine Corps to build user stories and use cases" [54]. The same consultant then said that they conduct operational demos after each sprint review to show off work done in the sprint and garner more feedback from the Marine Corps.

These use cases and user stories must then be translated into formal requirements in CDDs for formal contracting and funding allocation. The language within the CDD will be specific, but not exact. For instance, one engineer stated that these requirements must be similar to "Will build a graphical user interface," "Will do test and demonstration," or "Will build data feedback infrastructure" [55]. As one program manager at NAVAIR recalled, this was one of the chief problems during the AIM-9X project which was designed to upgrade the Sidewinder air-to-air missile [50]. The engineer went on to explain, during the early development in the 1990s, the software and processor could not adequately handle the computational load placed upon them by the sensors of the missile. Due to the way the JCIDS process works, the processor hardware, software packages, and even programming language all had to be specified in the missile's CDD. As the development program progressed and the computational hardware and software matured or became out of vogue, the CDD and other requirements documents had to be changed and go through the formal approval process multiple times. This led to repeated administrative delays and wastage of government funding and resources to perform these requirements reviews [50]. Reducing the specificity of requirements to the end goal of what is to be achieved provides the engineers and developers room to utilize the fullness of their creativity and leverage the latest technology to accomplish their design goals. As one contractor at PEO IWS exclaimed, "As long as the required operating capabilities (ROC) and planned operating environment (POE) for the ship and/or combat systems remains the same" then the developers should be able to add functionality and improvements as needed and shift the mindset from "we will add capability" to "we're going to make the system 'work better"' [15].

4.2.2 Model-Based Systems Engineering

But CDDs and other requirements documents are not the only instances of system documentation that must be updated. To deal with the issue of system complexity, the Navy must transition to the use of completely digital design methodologies and documentation. The chief amongst these is the use of model based systems engineering (MBSE). MBSE is "the practice of developing a set of related system models that help define, design, analyze,

and document the system under development. These models provide an efficient way to virtually prototype, explore, and communicate system aspects, while significantly reducing or eliminating dependence on traditional documents" [76]. MBSE allows engineers and developers to better create traceability throughout the entire system from initial concepts and requirements to final system designs and testing. For instance, as one contractor at NAVAIR noted, the MUX Program uses MBSE to specify requirements via "use cases within the MBSE model" and then "places performance constraints within the model" for developing test cases and system component designs [54]. This means that during any design review, the IPT can take any aspect of the system's design and trace it all the way back to the original capability request made by Sailors or the TYCOM advocate.

Another benefit of MBSE is that it provides for digital simulation of system behavior prior to any physical fabrication or coding is performed. This will help engineers, developers, and testing personnel to better understand complex systems interactions and interface requirements early on and assist with configuration management. It will also help the IPT to deconflict competing requirements early in the design process, saving time during systems integration testing and fabrication.

4.2.3 Open Systems and Configuration Management

Along with less specific requirements and the use of MBSE, the Navy must also rethink how it designs combat systems, or rather, how it requests for those combat systems to be designed by contractors. One consultant at NAVAIR best explained this by stating that "in DevOps the non-functional requirements are just as important as the functional requirements" and you must specify the usage of open standards and designs [54]. What is meant by "open" standards and designs is that the technology used conforms to third party specifications that are used commonly throughout industry. These standards can govern everything from nuts and bolts to programming languages and central processor architectures.

The goal of using open standards is to break the cycle of Navy systems being locked into only a handful of DoD contractors as discussed in Section 3.1.2. These open standards will make it easier to integrate new technology into existing combat systems, adding new functionality and making those combat systems more lethal. As explained by an assistant program manager at PEO Soldier, this was the goal of using open standards in the Army's

Adaptive Squad Architecture (ASA) Program [75]. The goal of the ASA Program is to treat the entire soldier as an integrated weapons system, similar to the way the the Navy treats ships and submarines, and then making continuous upgrades to the hardware and software that the soldier takes into battle with them [77]. That same assistant program manager stated, "The problem we are trying to solve is how do we deploy new systems in pieces so that we can field less equipment but more frequently" [75]. The Army found its solution by relying upon open designs such as ARM-based electronics and commercial wireless data standards [75] have paid dividends including allowing developers to implement brand new code improvements to the Army's new augmented reality headset in the field during the middle of a training exercise [78].

These open standards should be pulled from organizations such as SAE International, ASTM International, and the USB Implementers Forum as much as possible. These standards organizations provide third party guidelines for system components and interfaces across industries so that disparate systems can still be integrated together. As one contractor working on UAS systems stated, the non-functional design standards should not be "Will use 5/8 inch, stainless steel bolts" or "Will pass all data communication through port 8800" but rather should be "Will use SAE fasteners" and "Will use OMS/UCI data standard" [54]. This same contractor expanded upon this topic by saying that this use of open standards should by used throughout all aspects of the CDD and MBSE model.

4.2.4 Software Development

With the use of open standards, user stories for requirements, and MBSE modeling, the Navy will then be able to implement DevOps development. This is modeled upon the continuous integration model shown in Figure 2.2. This model attempts to explain the infinite cycle of DevOps in a linear fashion easily graspable by the human minds. This continuous integration cycle will be easiest to apply to Navy software as it was initially designed for software development. After modifying the nominal continuous integration cycle in the specific context of Naval combat systems, the process is diagrammed in Figure 4.5.

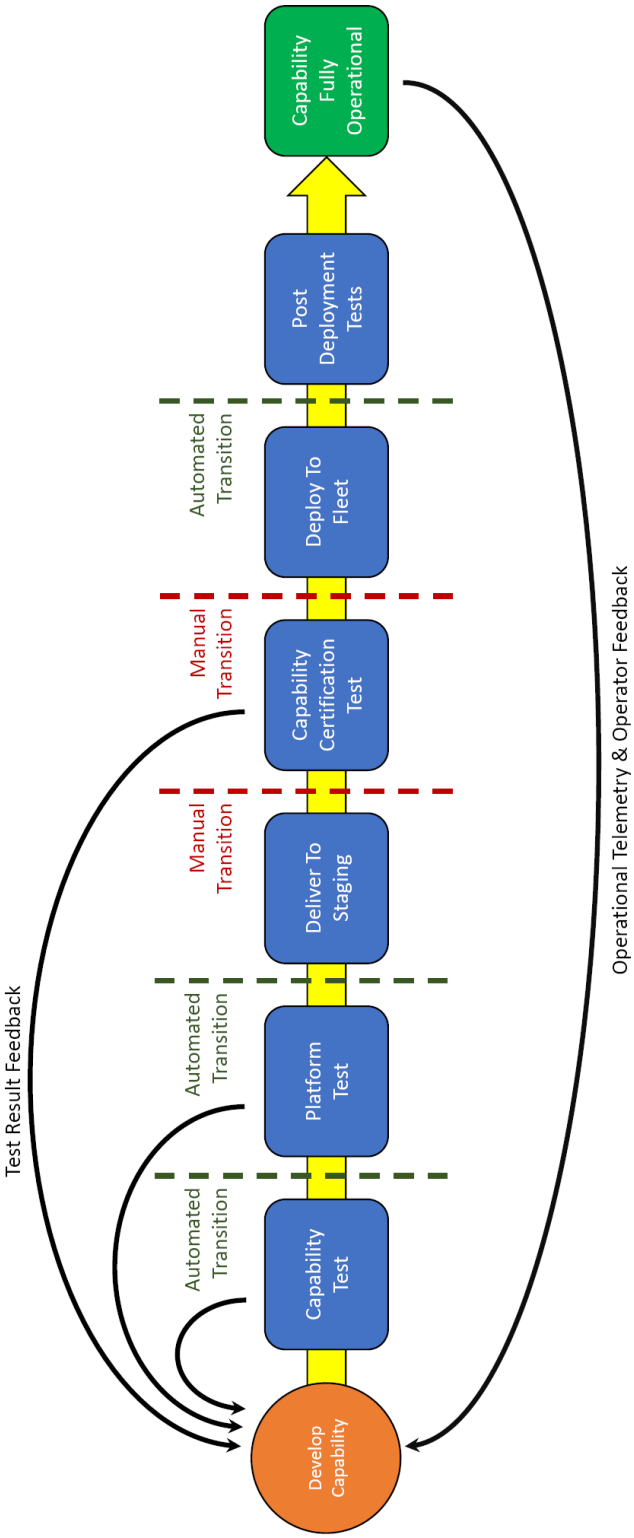

Figure 4.5. Navy Software Integration and Certification Path.

The cycle in Figure 4.5 begins with the initial development of some capability or functionality by an IPT based upon the requirements captured in user stories, the CDD, and MBSE model. This capability is some amount of functional code that could be a brand new piece of software or an upgrade to existing software. Once it has been developed to a functional point, then it will be automatically pushed into a testing environment where it will undergo functionality testing (Denoted as Capability Test in Figure 4.5) and then integrated testing as part of a larger combat systems suite such as Aegis or Ship Self-Defense System (SSDS) (Denoted as Platform Test in Figure 4.5). During both of these periods of testing, the resultant test data is fed back into the design of the software and improvements are made.

Once the software has gone through enough revisions, it will be readied for certification testing by COTF and AO personnel. Any failures during this testing and all performance data will be fed back into the design of the software making improvements and fixing defects. Once these defects have been fixed and the certification has been approved by TA and AO personnel, the software will be deployed to the fleet. After it has been deployed, post installation testing will be performed to validate the install and verify that the entire combat system suite is still functioning properly. This is already currently routinely performed at the end of new software and hardware installs with the use of Systems Operation and Verification Test (SOVT) and Combat Systems Ship Qualification Trials (CSSQT) testing to demonstrate that the newly installed systems are operating properly, the crew is properly trained to use them, and all of the necessary infrastructure is in place to support the new systems. Once all testing is complete, the software will relay operational telemetry and Sailor feedback for further improvements on the next update of the software.

Because of ship schedules and the requirements for the SOVT and CSSQT testing, this will likely limit the frequency with which new systems can be installed and updates can be pushed. While discussing this, one engineer stated, “The Submarine Force has one of the best programs with their Advance Processor Build Program where they refresh each boat every time they come into port for maintenance” [55]. The engineer further explained that this resulted in a roughly eighteen month update interval for each submarine with about two to four different combat systems configurations in the fleet at any given time. Under DevOps, the update interval would likely be the same. No matter how fast the development may be, the risk appetite within the Navy for unproven combat systems and the limitations

of logistics will always restrict how quickly the Navy is able to deploy new technology.

4.2.5 Hardware Development

While software development lends itself to the adoption of DevOps, hardware is another matter entirely. Unlike software, hardware has a slower development cycle due to its need to be built in the physical world and logistical lead times waiting for raw materials to arrive. This means that testing and systems integration cannot occur until prototypes have been manufactured and delivered. Depending upon the complexity of the system being developed, this could result in development taking weeks or months for prototypes to be fabricated. This is another area in which the use of open standards and commercially-available components can speed up the development of combat systems.

But that development can only be sped up so much. Because of this, the traditional two week sprints used within Agile Methods will not be adequate, just as Garzaniti et al. found [79] when they applied Agile Methods to the development of CubeSats which resulted in some of their sprints lasting a month or more. When Saab integrated Scrum and Agile Methods into their development of the Gripen fighter, they quickly found that the their optimal sprint length was three weeks [68]. One of the other lessons learned from Garzaniti's team [79] is that sprints must be planned based upon the time development tasks are expected to take instead of the traditional Fibonnaci story points method.

Another limitation to traditional Agile or DevOps development presented by hardware is that, due to the increased integration of software control into hardware systems, there must be complete synergy between hardware and software development during the construction of the initial prototype or model [80]. For modern complex combat systems, this means that the initial deployment of hardware cannot be delivered without the necessary software to operate it. In follow-on upgrades, software and hardware can be updated asynchronously but the foundation must be laid during the initial development.

Despite the limitations to Agile Methods and DevOps imposed by hardware, the implementation of Scrum, sprints, and other processes bring many benefits. One of these benefits that Garzaniti's team [79] found was that the frequency of the Scrum meetings aided with providing a regular tempo to development ensuring that the team stayed focused on meeting the project's end goals and did not become bogged down in extraneous tasks. Since the

team met so frequently, they were also able to respond quickly to requirements changes or to unforeseen events and delays, pivoting work priorities rapidly to respond to these delays. Taking these lessons into account, the continuous integration cycle must be modified for hardware development. This new cycle model is detailed in Figure 4.6.

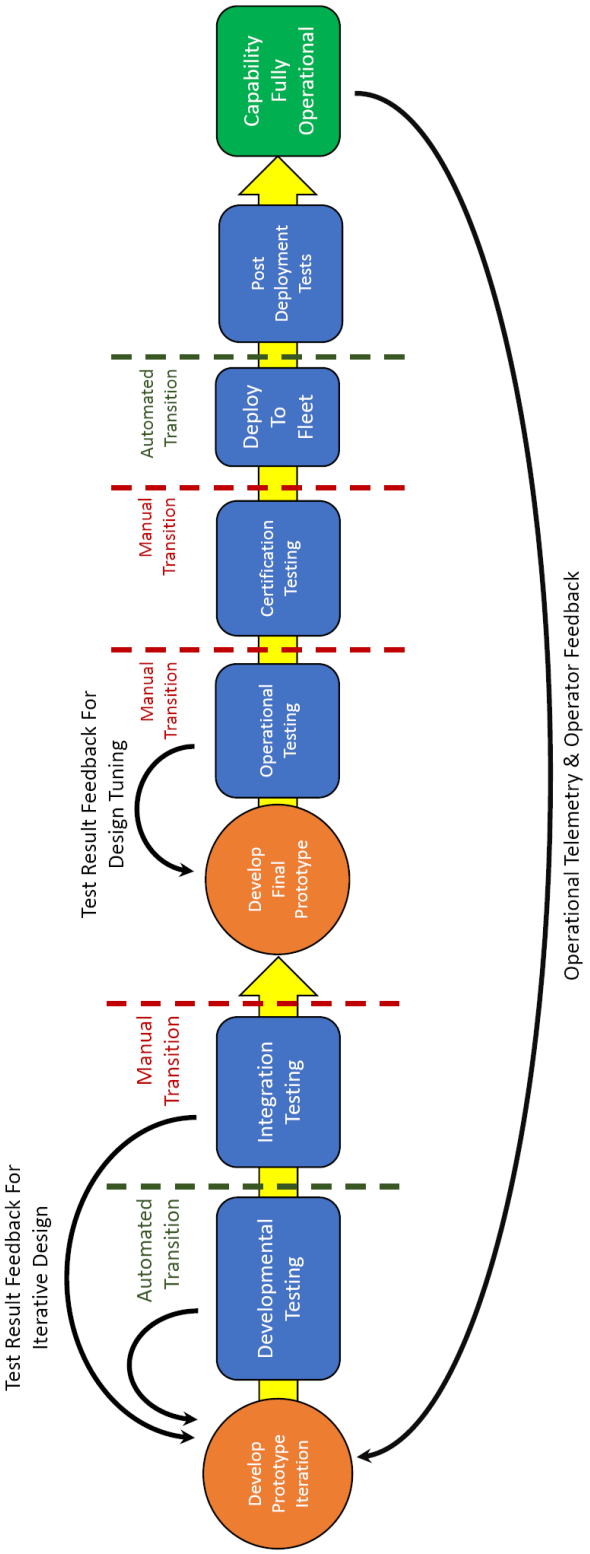

Figure 4.6. Navy Hardware Integration and Certification Path.

Like the software cycle, the hardware cycle begins with the initial development of some system prototype by an IPT based upon the requirements captured in user stories, the CDD, and MBSE model. A components are fabricated they are sent through developmental testing to prove design concepts and better understand their predicted behavior in an integrated systems context. As mentioned earlier, the wait for fabrication of prototype components can delay the start of testing considerably. To shorten this wait time, as a program manager at NAVAIR explained, the Combat Dragon Program "used a 3D printer to provide rapid prototyping of new system components as well as even replacement components which had broken" and had a long lead time for delivery, delaying testing [50], [81]. From literature, Garzaniti's team [79] also made extensive use of additive manufacturing and other rapid prototyping techniques to quickly fabricate prototypes for mechanical and circuit board components so that they could begin systems integration and testing sooner.

The results of the development tests will then be used to refine the MBSE model and prototype designs. These revised prototypes will then be integrated into the overall combat system for integrated operational testing (Denoted as Integration Testing in Figure 4.6). This integration testing will be used to detect negative component interactions and begin the refinement of the system's design to reach the needed level of operational capability. At some point during this development period, as the combat system reaches a certain predetermined level of maturity (Decided upon by consensus within the IPT), all requirements will be frozen in place and no new functionality will be added. This decision will create a baseline for the final prototype design (Denoted as Develop Final Prototype in Figure 4.6) and must be performed manually after a design and progress review by the IPT and program office. This baseline creation is crucial as it prevents constant change and an endless cycle of development with no final product [82]. This is echoed by a response from a program manager at NAVAIR who stated, "Two big problems that kill any program are requirements creep and work spiral. To prevent this, every so often we need to create 'mini-freezes' where we decide this is how up to date we will be since we can't always have the latest and greatest technology" [50].

This final prototype will be put through operational testing to find bugs and troubleshoot integration problems. The data from this operational testing will be used to tune the prototype and update the MBSE model to prepare the design for production. Once the prototype is deemed ready, it will be handed to TA and COTF personnel for certification testing. Once

it passes this testing satisfactorily, it will be put into production and deployed to the fleet. Much like software, this new combat systems hardware will be tested one final time through the SOVT and CSSQT process to ensure that it is functioning on each ship or submarine as designed. It will then operate at sea, generating operational telemetry and user feedback which will be fed back to the development IPT for the purpose of improving the next design iteration, beginning the cycle anew.

4.3 Faster, Better Testing

Even with changes to the way the Navy develops combat systems, it will still need to meet the regulatory requirements to prove that those systems are safe and effective. As one contractor at PEO IWS stated, "Testing is all about proving to TA and COTF that we know what we're doing" [51]. The DevOps approach to testing will likely accomplish this far better than the infrequent, spasmodic testing performed in conjunction with project milestones under the DAS.

In order to accomplish this testing, two criteria must be met. First, multiple testing environments must be established in a manner similar to those described in Section 2.1.1. The second is more difficult as it requires separating functional and acceptance testing [36]. As explained in Section 3.3.2, the Navy already divides its testing into functional tests (DT&E) and acceptance tests (OT&E) but that means that combat systems are often not tested in an operational environment in a non-punitive manner [15]. To remedy this will require a change in culture so that integrated testing can be conducted in an operational environment so that valuable lessons can be learned and incorporated into the system's design.

4.3.1 Multi-Stage Testing Environment

To facilitate this change in testing, each development program will need to establish separate testing environments to achieve the increase in development velocity discussed in Section 3.3.2. These environments will need to be configured nearly identically so as to provide the environment parity discussed in Section 2.1.1. By creating these multiple environments, the Navy can test the software and hardware systems in development more often and more consistently. There will need to be four environments in total, which allow for developmental, operational, and certification testing necessary to prepare combat systems for use in the fleet.

The first and most important environment is the fleet environment. This is the combat system operating aboard ships and downrange in the hands of Sailors. The fleet environment is analogous to the production environment in private industry. A combat system or its components can only reach this fleet environment after being tested and certified by the TA, AO, and COTF personnel.

The second environment is the certification testing environment. This certification environment is immediately upstream from the fleet environment and would be analogous to the current OT&E that is performed now. In the certification environment, real world conditions and systems are recreated as accurately as possible to provide the closest real world conditions as possible. This would involve building out a shipboard analog to integrate new components into and then push them through rigorous tests to prove their combat effectiveness and safety. This would be similar to the "Cruiser in the Cornfield" that exists at the Aegis Technical Representative offices in New Jersey. This "Cruiser" is a building with a ship's superstructure, combat systems housings, and antennas replicated on top of it. In its interior are replicas of the exact combat systems control rooms that exist on board ships at sea. By leveraging similar environments, the Navy could conduct integrated certification testing without having to first install the system on board a ship and then test it there as is currently done [51]. But no matter how realistic this certification environment is, there will still need to be quality control testing performed once the combat system has been installed aboard a ship. According to one consultant at NAVSEA, this would be accomplished through the Navy's current SOVT process which ensures that combat systems perform as expected once they have been installed on the ship and that the crew is properly trained to operate the new or upgraded combat systems [15].

The remaining two environments exist for what would currently be called developmental test and evaluation (DT&E). These environments exist to provide, as one engineer state, "a non-punitive environment for testing so that developers can test their ideas in a sandbox before integrating them into the final product" [55]. As stated in Section 2.1.1, the ultimate goal of these development environments is to "break" the combat system early and often to determine efficacy of designs and to find defects quickly.

The first of these development environments is for pure development. This is most analogous to the development environments used in private industry and provides synthetic inputs and

targets to test design ideas. For physical systems this would also utilize synthetic targets, threats, and weapons to determine functionality of the initial prototypes of the combat system components and subsystems. This environment is purely for proof of concept testing and fulfills the role of contractor-performed DT&E in the current DAS.

The next stage of testing is an integration testbed and is immediately upstream from the certification environment and is for performing integrated operational testing prior to the certification testing so that the combat system can be fully assembled and assessed for effectiveness, technical readiness, and troubleshooting or debugging. This will rely upon synthetic inputs with real world inputs added as practicable. For instance, if a radar system is being tested, it may be tasked with tracking real aircraft but the missiles fired at it are digital. Furthermore, if that radar is integrated into an entire combat systems suite, then the control software such as Aegis is being run on regular server hardware and is sending commands to a real vertical launch system that is loaded with digital missiles. This would provide the means for the developers and PA personnel to better understand how far along the combat system is in development as well as understand how well it will interact with other systems when integrated on a ship so that they can perform troubleshooting, debugging, and design revisions. As one engineer described this new integration testbed, "The sandbox is pond where a fish can grow to 12 inches, the testbed is where it can grow to 24 inches, and then you release it into the wild [Certification testing]" [55].

4.3.2 Software Testing

Even with it explained, it is difficult to see this new testing paradigm out of context. Applying this new multi-stage testing process to the development of software, provides a clearer picture of how the process will function and the way the development discussed in Section 4.2 will flow. Since software can be developed asynchronously from hardware, this new multi-stage process will look and function in a manner most similar to the development and testing performed within private industry.

The new multi-stage cycle is shown in Figure 4.7. In this diagram, the entire life cycle of a software development program is outlined from its beginning in the minds of the personnel within a capability IPT, through the multiple stages of testing, and then finally out to the fleet. Throughout this entire process, data is generated and fed back to the IPTs personnel

for design and code revisions. One thing to note is that the vast majority of DT&E and initial OT&E will be automated to reduce the human workload on developers and testers as well as allow for a greater volume of testing in a shorter period of time.

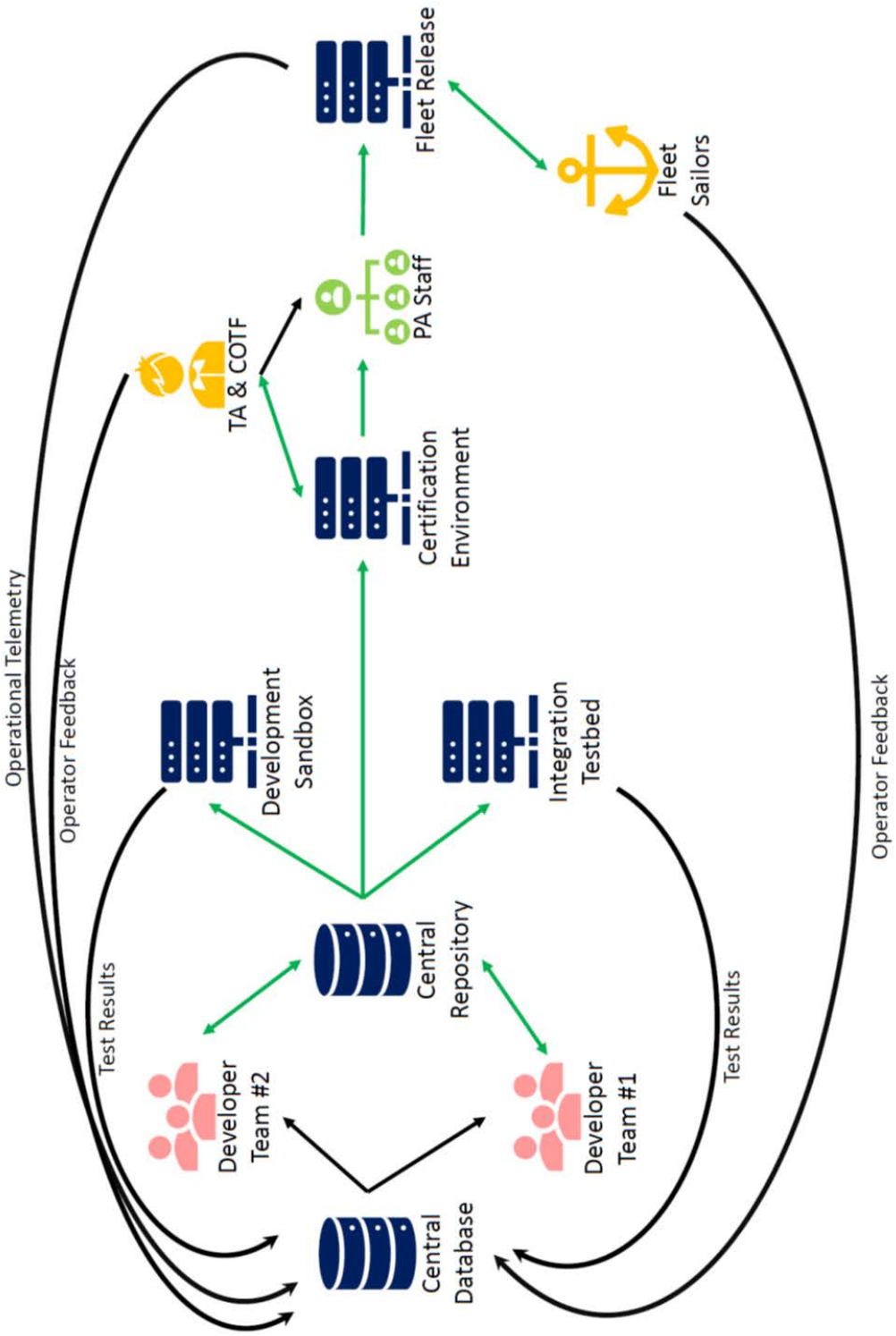

Figure 4.7. Navy Software Development Flow. Adapted from [36].

In Figure 4.7, the development process begins with an IPT (Shown in pink) determining requirements with embedded TYCOM advocates and building an MBSE model as discussed in Section 4.2.2. They then take this MBSE model and convert it into lines of code. These lines of code are then committed into a central repository similar to GitHub which allows for multiple developers to work on bits and pieces of the final software system and then compiled those bits and pieces into a functional product. This central repository becomes the standard for work and progress on the software.

Once a sufficient amount of development has been completed, the developers push it into the development sandbox where they can conduct functional testing and test ideas to prove concepts before committing them to a release candidate. This process mirrors the testing of alphas or nightly releases in private industry. In this development sandbox, the developers will not only perform functional tests but will also conduct fuzz testing using applications similar to Netflix's Chaos Monkey software. The Chaos Monkey software injects non-design failure modes and use cases into the testing to find unforeseen problems [83]. Not only does fuzz testing inject failure in ways planned by humans, but also in ways that humans can't predict. Not only does it do this, but it is also automated, allowing developers to run tests overnight when they are not physically at work.

This automation allows the developers to come in the next morning and to analyze the data generated from the testing to then fix defects and refine their design. After several iterations of this development testing, the developers will solidify the software system into a release candidate, similar to a beta release. This release candidate will be an approximation of the final software system and will be pushed into the integration testbed for operational testing as part of the rest of the combat systems suite it is being developed for. This will still rely heavily upon automated synthetic functional and fuzz testing but will also begin to incorporate real physical tests and human users. These tests will be designed by TA and COTF personnel embedded into the IPT so as to prepare the software for the final certification testing. This integrated testing will also allow the developers to begin to find operational defects as well as better understand human requirements for the software. This will also flush out integration hurdles as part of the need to perform continuous integration.

This integrated testing will allow for further refinements of the software into a finished product before it is pushed to the certification environment where TA and COTF personnel

from outside of the IPT will conduct final operational testing to ensure that the software can perform its duties and that the entire combat systems suite can still maintain its combat effectiveness and safety. By using testing personnel from outside the IPT, the testers can still maintain the objectivity and freedom from conflicts of interest that they require to meet statutory and ethical standards for certification. This certification testing will be performed in an environment that replicates the real shipboard configuration as possible. For instance, as one consultant relayed, PEO IWS is in the process of building out a facility in Virginia which entirely replicates the combat information center, weapons systems, sensors, and command, control, computers, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) systems for the SSDS combat systems suite [51]. This will be able to fire live ordnance and perform physical intercepts of targets. This will be able to simulate for TA and AO certifiers adequately without having to work around ship schedules, causing program delays.

But this testing profile isn't just for new software, but also for updates. This certification environment will test updates to existing software and ensure that those updates do not cause malfunctions or degradation in the efficacy or safety of the combat systems suite. This will allow for certification of just those updates. These certification tests will also generate data that is fed back to developers for further refinement of the software in the next release.

Once the software has satisfactorily passed the certification testing, the PA staff will prepare all of the necessary documentation and training and issue it to the fleet prior to delivery. The release candidate will then be transitioned to a fleet release and pushed out to the fleet. This fleet release will be issued over time so that not every ship is updated all at once. This prevents the Navy's entire combat capability from being rendered impotent by unforeseen malfunctions. This will likely result in software updates being pushed out strike group by strike group or fleet by fleet. Once the software has reached fleet release, it will generate operational data that is then sent back to the IPTs for further refinement. Not only will the IPTs receive telemetry data but they will also receive direct feedback from Sailors through the submission of trouble reports and feature requests. All of this data will then be compiled, analyzed, and used to justify the requirements for the next release during Scrum meetings as the cycle restarts.

4.3.3 Hardware Testing

With the understanding of how software will be developed and tested, the next step is to discuss the cycle for hardware. As discussed in Section 3.4.1, the Navy does not have the luxury of developing software in a vacuum and must also have a way to develop hardware. But the development of hardware has its own peculiarities which will force the need to adapt to a slightly different testing cycle as shown in Figure 4.8.

The biggest difference is that the need for a testbed will be much greater since integration will not just involve lines of code but also physical interference, electrical supply, and management of the electromagnetic spectrum. Also, the reliance upon automation will likely not be so great. Because the Navy designs hardware for use by humans, human testing will have to be integrated much earlier in the process than with software. This will include testing for usability and maintainability. This will slow the pace of hardware development, which is one of the reasons that hardware acquisitions programs will still likely be slower than software in a DevOps acquisitions system.

Like the software cycle in Figure 4.7, the hardware cycle begins with personnel assigned to a capability IPT determining requirements with the help of embedded TYCOM advocates and then building an MBSE model based upon those requirements. This model will go through several refinements before any work on physical hardware begins. One of the benefits of MBSE is the ability to perform rudimentary simulations within the model to better understand system interactions [84]. This will allow developers and engineers to be better informed when they begin designing system components and to predict overall system behavior so as to prevent negative interactions in their designs.

These component designs will need to be performed using computer-aided design so that they can then be digitally linked back to the MBSE model for traceability. With these digital designs, contractors will fabricate and assemble system components for testing within a development sandbox. This sandbox will consist of functional and quality control testing to prove design concepts and manufacturing processes. This developmental testing will rely upon automation as much as possible in order to get results data back in the hands of engineers as possible. The more data that can be fed back into the design process and the faster and more often it can be fed in, the faster the pace of hardware development can be achieved.

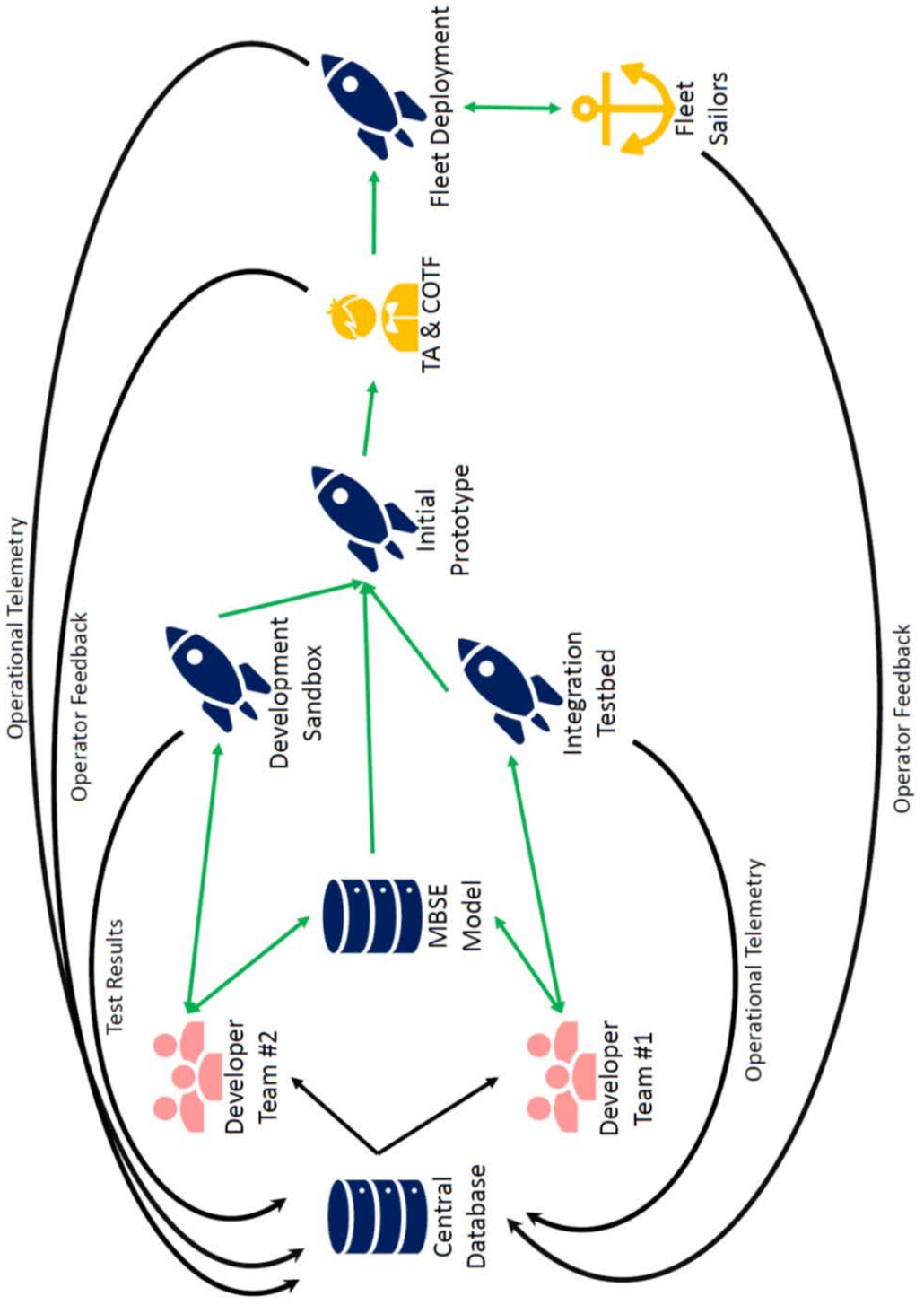

Figure 4.8. Navy Hardware Development Flow. Adapted from [36].

Each piece of developmental testing data will be used to refine the MBSE model and digital designs. Once the model, designs, and prototypes reach a certain technology readiness level (TRL), they will be integrated into an integration testbed. The specific threshold to move onto integrated operational testing will be determined by the specific program office and PA personnel. This should be done as soon as practicable in order to begin learning integration lessons early in the system's life cycle sooner rather than later. Once again, data from these operational tests will be used to refine the MBSE model and digital designs.

It cannot be overstated how important this integration testbed is for hardware development. Without a way to test integration processes and see how the fully integrated system will perform, there may not be any way to find out prior to certification. As one program manager at NAVAIR recalled, they had "learned this lesson the hard way" during the Combat Dragon program and were forced to devote an entire aircraft to act only as an integration testbed to ensure subsystems and system components could fit inside of the aircraft and still work with the other aircraft systems on board [50]. Similar experiences have also driven PEO IWS to pursue building multiple systems for the SSDS program. As one contractor explained, "Testing will have three sets of hardware: one set in the development office, one set in Virginia for testing, and one set on each ship. You perform development with one set of hardware with simulated sensors and then you take SSDS components and fit them to actual sensors to test functionality" [51]. Again, the use of integration testbeds and developmental sandboxes will need to be the norm for combat systems development under any DevOps system in order to achieve the goal of continuous integration.

The data generated by this integrated operational testing will be used to refine the MBSE model and digital designs and once sufficient integrated operational testing has been completed the PA, TA, and AO personnel deem the system is ready for certification. When that decision is made, one or more initial production prototypes will be built based upon this refined MBSE model and handed to TA and COTF personnel to test under real world conditions. As in the software testing cycle, the TA and COTF personnel who perform the certification will be different from those embedded into the developmental IPT so as to avoid conflicts of interest. Upon completion of this testing, test results will be used to make any necessary configuration adjustments and the hardware will be put into production and deployed to operational units.

Once installed on ships, submarines, or aircraft, those systems would be placed through verification testing such as SOVT or CSSQT tests to ensure that they were installed properly and the crew was able to operate them effectively and safely. Those same crews and operational units will provide operational telemetry and formal user feedback through trouble reports and feature requests, allowing developers and engineers to better understand how the system was employed in the real world. This would then allow them to incorporate those changes into the next prototype and baseline system, beginning the cycle over again.

4.4 Getting the Right Information to the Right People

As discussed in Section 2.1.3, one of the chief principles of any DevOps system is the incorporation of data feedback from operations and testing back into the development process. Currently, the Navy does not have the infrastructure in place for the collection and distribution of this as examined in Section 3.4.2. In order to transition to a DevOps acquisitions system, the Navy will need to realize the value of data collection, analysis, and sharing.

4.4.1 System Instrumentation

The first step of any operational feedback infrastructure is the physical collection of data. A software engineer who works for NAVWAR explained that it is crucial "that we configure each system to collect everything a Sailor does and report it back" to the program office and developers [28]. While this is not an entirely foreign concept to the Navy, as explained in Section 3.4.2, the Navy requires deliberate planning, the use of external recording equipment, and manual system configuration to collect and store test data. The biggest difference must be that instead of aligning this data collection with milestone testing, it must be a natural state of each component and subsystem within any combat systems. Pursuant to this goal, the requirements for data collection, storage, and management must be written into any contracts of capabilities documents for each program.

This data collection is performed using instrumentation of the software or hardware components. As the same aforementioned NAVWAR engineer mentioned, the "move to digital systems" has allowed "for the automated tracking of state changes and collection of data" [28]. This jives with the practices within private industry. For instance, Microsoft uses

software instrumentation to collect data by both tracking state changes of the software and performing synthetic transactions to measure software performance [85]. These synthetic transactions pass empty data packets and commands throughout the system to measure speed and other parameters.

4.4.2 Storing and Supplying Data

But it is not enough to simply collect data; that data must then be stored and distributed in order to be useful. As explained in Section 2.1.3, one of the biggest components of any DevOps system is the incorporation of data from operating systems back into the design of the original system. To do this, the Navy must satisfy the security and classification restrictions imposed upon it due to the Navy's existence within an HRE. Any infrastructure must place limits on who can access what information and those actions they can take with that information.

This is commonly done within private industry using a "zero trust" model of software and database access [86]. In this "zero trust" model, access is only granted to resources and data to those personnel who are properly vetted and have a need for that access. This is very similar to the siloing of information that is performed within the DoD but in a purely digital environment. When collecting large amounts of data, most private companies utilize relational databases such as MySQL or MariaDB that reside on a central server and are accessed automatically by various software [87]. These databases have built-in user management systems that are capable of restricting access based on users' privileges. These databases can also be easily replicated on different servers to maintain integrity of the data on the original server [88]. This replication can be performed one-way or synchronously and can be a partial or full, which can be done to satisfy security requirements.

By applying this "zero trust" model and leveraging the capabilities of modern relational databases, an effective process architecture can be constructed. This process for providing instrumentation data and fleet feedback is shown in Figure 4.9. In this architecture, the rocket ships represent deployed combat systems and the pink people represent the development IPTs for capabilities for that combat system. The databases containing the feedback data are represented by the blue cylinders.

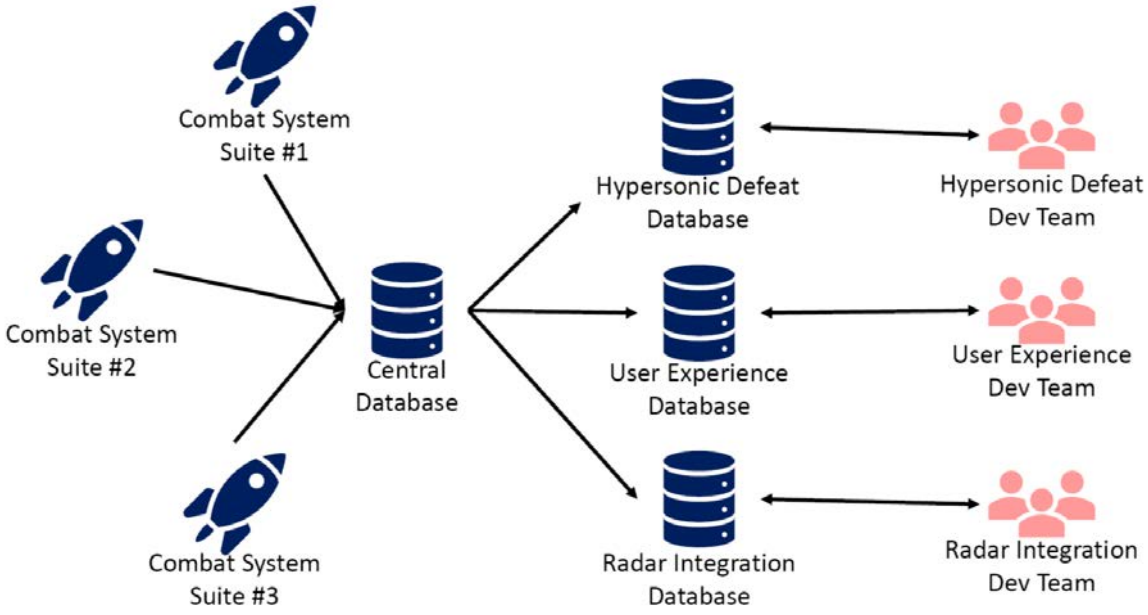

Figure 4.9. Navy DevOps Feedback Loop.

In this feedback process, the instrumentation data and user feedback would collected in a small database on board the ship or submarine. Those shipboard databases would then send their data to a central, secure database located somewhere at the program office and managed by the PA. This data transfer would be encrypted and secure using standard encryption practices and encrypted connections such as virtual private networks (VPNs). And since data connections at sea are not always stable, the partial, asynchronous replication of databases allows for the data to be sent in pieces back to the central database [87].

The central database is the most secure portion of this process. Since it will contain performance and operational data from every version of a specific combat system in operation, it will need the utmost most security. This central database will also reside at the highest security classification since data aggregation can often lead to an increase in classification levels. This will result in strict access restrictions, both physical and digital, much password or private information databases in private industry [63]. To still allow the development teams to access necessary information, portions of this central database will be replicated via a one-way process in smaller distributed databases located with the development teams. This can be performed automatically or manually. It can even be performed by transporting

physical media from the central databases to one of the development databases in case the central database needs to be "air gapped" for security reasons [63].

These distributed databases will contain the data that each development team needs to perform their work. Continuing the same example from Figure 4.4, the hypersonic threat team will have different data than the team working on the user experience. Also, since the hypersonic threat team will have information regarding enemy missile performance it will also likely exist at a higher classification level than the user experience team's database that only contains data regarding the user interface. This will continue the siloing and partition of information mandated by statute to safeguard state secrets based upon need to know. The distributed databases also allow for the development teams to modify data without disturbing other teams' data. This will give the data scientists within each team (as discussed in Section 3.4.2) to perform analysis and make modifications as necessary to elucidate design requirements and performance shortfalls as discussed in Section 4.2.

4.4.3 Making Decisions Based Upon that Information

With the aforementioned feedback process and distributed databases in place, the Navy can provide far more information to TA and AO personnel for them to make their certification and ATO decisions. But the shear amount of data that will be generated and collected will need analysis by trained data scientists as discussed in Section 3.4.2. These data scientists will need to be embedded directly into each IPT so that they can work closely with the rest of the team to find answers or better elucidate questions brought up by operational needs, direct operator feedback (e.g., user feedback reports and trouble reports), and sub-optimal system performance during testing.

Armed with the capability to perform deep data analysis and having decouple testing events from acquisitions milestones as discussed in Section 4.3, each IPT can better answer the real behind the DIACAP and combat systems certification processes: What are the full capabilities and limitations of our combat systems when integrated into the full shipboard system of combat systems? This understanding is what will allow the TA and AO personnel to certify the incremental improvements of combat system software and hardware. This idea is behind the efforts at PEO IWS to build an element certification policy [89] that allows for radars or software to be added or upgraded and certified on a particular combat systems

suite without having to re-certify the entire suite.

As one manager involved with the combat systems certification process explained, DevOps requires "that we better understand the overall system's performance when integrated" as part of an entire ship's systems [61]. This manager further clarified, "Essentially, we need to learn how to deliver partial warfare capabilities or understand the full extent to our ability to perform warfare tasks such as air and missile defense or surface warfare." This represents a paradigm shift from the current certification model in which TA and COTF personnel begin with a warfare area (e.g., Undersea Warfare) and then work from the top downwards to determine which systems must be tested and what specific requirements must be met to prove that the system and ship can participate in that warfare area [90]. For example, to certify a ship as being able to perform air defense, the TA and COTF personnel would determine that the vertical launch cells, missiles, radars, and Aegis software would need to meet specific performance criteria such as being able to launch missiles in a certain time, track targets within a specific precision range, and detect targets within a specific time. Based upon the test results, the TA then makes a yes or no decision on overall combat system certification while capturing any degradation or limitations in discrepancy and trouble reports [90]. As discussed in Sections 3.3.2 and 3.3.3, this leads to a cumbersome process that doesn't necessarily demonstrate that a combat system can actually perform needed operational tasks.

Under the new DevOps paradigm, the TA and AO personnel would be able to review months or years of DT&E and integrated OT&E data prior to actual certification tests. This will allow them to better codify and understand the combat system's readiness for deployment using a process similar to TRLs. Coupling this better understanding of system readiness with the direct input of needs from the fleet via TYCOM advocates as discussed in Sections 4.1 and 4.2.2, certification can better capture partial capabilities and explain them to the operators in the fleet. This will result in certification criteria changing from "Can the vertical launch system launch a missile in 0.5 seconds from button depress?" to "Can the overall combat systems suite successfully intercept and destroy a subsonic aircraft?"

But what does this need decision process look like? As shown in Figure 4.10, the RMF process discussed in Section 3.3.3 can be adapted into a new Continuous Certification Framework. This transforms the certification process from a linear affair into a continuous cycle.

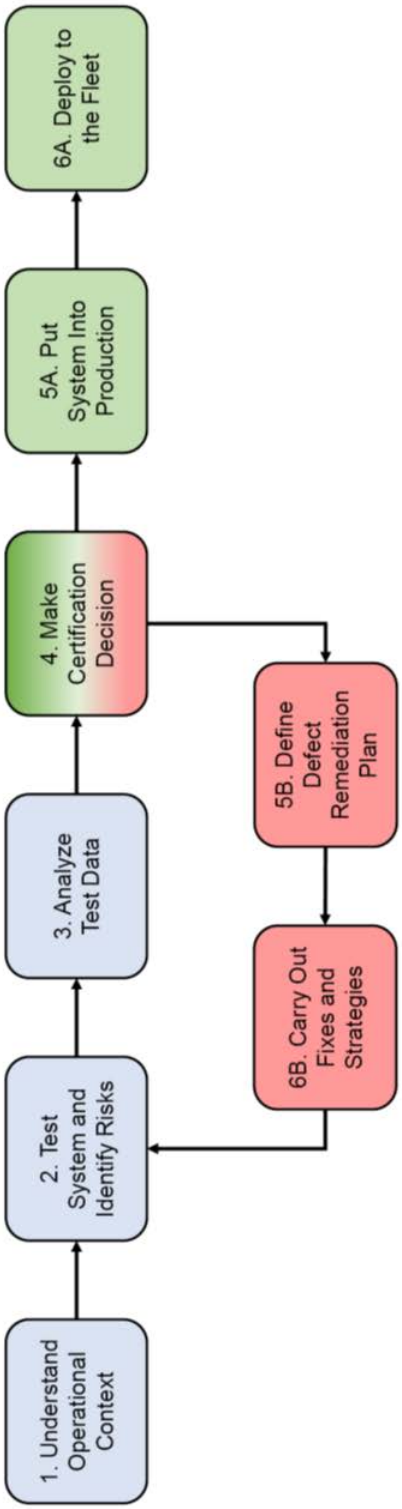

Figure 4.10. Navy DevOps Continuous Certification Framework. Adapted from [70].

To explain this new process in context, imagine that there is an IPT at PEO IWS who are responsible for integrating a new radar into the SSDS combat systems suite. This radar IPT would perform development, working the new radar through the testbed and integration sandbox environments as discussed in Section 4.3. This will provide data that will allow the TA and COTF to understand the radar's readiness for final production and deployment to the fleet and how it will perform when integrated with the rest of the SSDS suite. This will allow them to then develop holistic testing that demonstrates not only how accurately and precisely the radar can detect targets but also if the entire SSDS suite still detect, track, and intercept enemy aircraft and missiles once the new radar is integrated.

Putting this through the new Continuous Certification Process, the radar would be integrated into a certification testing environment, would then conduct testing, and the test results would be analyzed. Based upon the test results the TA or AO personnel would make a decision about whether to certify the new radar or not. This decision is the inflection point of the cycle and is the "Go" or "No-Go" (Hence the color graduation of block 4 in Figure 4.10). If the radar is granted certification, the PA and contractors will proceed with the production and fielding of the new radars to the fleet (Blocks 5A and 6A in Figure 4.10).

If the TA or AO personnel do not certify the radar then the PA and contractor personnel will develop and carry out a remediation plan (Items 5B and 6B in Figure 4.10) to improve the performance of the radar and its overall integration with the SSDS combat systems suite. This will incorporate sub-optimal performance back into the Scrum planning process so as to allow the TYCOM advocates and PA staff to reevaluate the priorities of the new radar project and overall SSDS program based upon the time and cost it will take to remedy the defects within the radar to meet certification criteria.

Another avenue that the TA and AO could pursue is to partially certify the radar. This would occur if, for instance, the radar were able to adequately detect and track subsonic threats but could not do so for supersonic threats. By communicating this partial certification to the operators in the fleet and Navy leadership, these decision-makers can make an informed decision if the added capabilities of the new radar are worth its limitations and whether they truly want to pursue fielding the radar in its current state. This would also better inform operators to develop tactics and strategies to ameliorate the limitations of the new radar.

4.5 More Frequent Delivery

After achieving certification, new combat systems must then be fielded to the fleet. As one software engineer remarked, "If you don't provide a product, DevOps is worthless" [28]. This sentiment was echoed by all other respondents in not as many words. At its essence, all of the efforts to transition the Navy's acquisitions strategy to a DevOps system will be for naught if it does not improve the speed at which combat systems are delivered to the fleet and the fit of those combat systems to the needs of the Sailors at sea.

That being said, there will be limitations to the benefits to speed of delivery that the Navy will see. All eight respondents who had experience working with the Navy expressed concerns that the Navy would ever see the same speed of delivery that private industry has been able to achieve (Sometimes measured in hours [54]). This is due to the risk averse nature of Navy leadership and its status as an HRO that was discussed in Section 3.1.1 and 3.1.3, respectively. The subsequent need to test and certify every piece of hardware and software to ensure it operates effectively and safely (As discussed in Sections 3.3.2 and 3.3.3) will inevitably slow down the delivery of combat systems to the Navy.

4.5.1 Software Delivery

Despite these limitations, the delivery of software is the most likely to benefit from the adoption of DevOps. The three personnel directly engaged in software development for the Navy have all mentioned that they already employ many of the practices of Agile and DevOps by breaking work into smaller pieces and then delivering these smaller updates more frequently. For NAVAIR, these updates are driven by changing needs and feedback from the customer, sometimes on a daily basis [55]. None of the parties expressed concerns ability of the Navy to eventually utilize automated software updates and deliveries via the Navy's existing intranet infrastructure and communications networks. One engineer even mentioned that smaller, more frequent updates would make the task of passing updates to ships with limited bandwidth easier due to smaller file size [28].

But even with this modernization of software delivery, there would still be limitations. One cyber engineer and a software engineer mentioned that there would be some systems that could not be incorporated into this new delivery system. Due to security concerns, some systems must be air gapped or physically disconnected from the rest of the internet or the

Navy's larger network [45]. Because of these security concerns, no matter how capable any automated delivery system becomes, there will still be a small number combat systems that will need software updated and delivered manually.

4.5.2 Hardware Delivery

Unlike software, hardware delivery will likely look much the same as it does under the DAS. The need to not only test and certify but physically build and develop the will be slow. Also, lessons will need to be learned with each system being delivered as kinks and defects are massaged out of the installation process. As three of the respondents remarked, this is due to the complexity and linear path of hardware development. As one cyber security engineer mentioned, "From my experience performing upgrades and installs on every ship class in the Navy, no ship as built has ever matched its design. We have always needed to write departures from specifications, made modifications, and submitted waivers when performing installs. This is because each ship is a complex system built by humans so it's bound to have errors" [45]. Another program manager echoed these sentiments that when working with aircraft systems they have often "had to treat the first aircraft as a test bench and incorporate those lessons learned" into the design of the subsequent aircraft [50].

But this does not mean that shifting to a DevOps system is without benefits for hardware development. By abstracting the development of hardware away from the development of software as discussed in Section 4.2, hardware would be able to be delivered independently from software or piecemeal instead as an entire combat systems suite. As one project manager at PEO IWS mentioned, DevOps would allow the Navy to deliver hardware in a manner similar to Infrastructure as a Service (IaaS) instead of the current manner [74]. As explained by a contractor at NAVSEA, the current process involves every component and subsystem in a combat systems suite being assembled, tested for functionality, disassembled, installed on the ship and then tested for certification [51]. As used in private industry, IaaS deploys server and network components such as servers and switches individually or in small groups as needed. By borrowing this idea, individual components and subsystems of a combat systems suite could be deployed to the fleet without the rest of the suite itself. For instance, operator terminals could be upgraded with new touchscreens or a new fire control radar could be installed without first having to assemble the entirety of the combat systems suite in a warehouse for testing. This would save time, money, and allow for more flexible

delivery of capabilities to the fleet.

4.6 How Will The Navy Change Its Culture?

Throughout the interviews for this thesis, it seemed as though much of the dysfunction in communication and collaboration between the various personnel in the Navy's acquisitions corps was due to a lack of trust between the various offices and personnel who make up the DAS. Many of the respondents mentioned antagonistic relationships between PA and TA and between the COTF and contractors. One contractor in particular noted, "We are going to get enormous amounts of push back and people are not going to like" the change [55]. But it's not just mistrust between humans that must be rectified. As explained in Section 3.1, there is also mistrust of the concepts behind DevOps as well as the combat systems it produces. This lack of trust is caustic to the open coordination and collaboration that is necessary for any DevOps system to function.

4.6.1 Building Trust Between Humans

To facilitate the need for transparency needed between teams in DevOps, trust must be built between humans. Trust is the knowing acceptance of risk on behalf of one person based upon an expected outcome guaranteed by another person [91]. For humans, this trust is built based upon communication (both verbal and non-verbal) and behavior. Time and resources must be dedicated for teams within the Navy's acquisitions corps to socialize and build group identities [59] that will allow them to interact smoothly with each other and ensure harmony during intra-group interactions. Part of this deliberate socializing should be inter-disciplinary and DevOps training. This training will bring together personnel from every office and every rank within the Navy's acquisitions corps and will focus on teaching personnel how to communicate and interact with each other and how each person is helping to work towards the shared mission of delivering capability to Sailors.

This training will reduce the personal distance between personnel and increase the intimacy level so that trust can form across all sectors of the acquisitions corps [59]. Fully integrated training for all personnel was one of the key lessons learned during the Compile to Combat - 24 Hours (C2C24) project executed by NAVWAR in 2018 [92]. As a manager who was involved in the project explained, NAVWAR found that all of the personnel involved in the

project had to be trained and educated on how to perform in an Agile or DevOps environment. Trying to train personnel in a piecemeal fashion as the Navy is used to doing caused too much friction and reduced the effectiveness of communication in the organization [28].

But training isn't the solution for building trust between the various personnel involved in the DevOps system. The Navy must also change the way it works in development commands as well as its expectations for what its acquisitions and contractor workforce looks like. Currently the Navy has a very conservative office culture that must be relaxed to foster collaboration across the entirety of the Navy's acquisitions corps. This includes relaxing dress codes and formality. For example, in one of the project offices of the Air Force's Kessel Run Program, the military personnel do not wear uniforms. As one of the project's managers explains:

> "We don't wear uniforms on a daily basis because it eliminates bias and fears of speaking up. It also breaks down the divide between enlisted and officers as well as between military and civilian contractors. We had an junior enlisted developer tell a flag officer what his ideas were and we later implemented those ideas. The whole point is to eliminate the things that hamper the free flow of ideas" [52].

That same manager also said that sometimes military uniforms are encouraged, typically when meeting with military customers who will be using the project team's products. As the manager said, it is "important for military personnel to see other military personnel working on our products" because it "builds trust with the operators" [52]. This is a fundamental shift in how the Navy does business but will be required to attract the personnel with the right expertise to maintain the Navy's technological edge. As one engineer at NAVSEA expounded, "Many of the most talented people in Silicon Valley have unorthodox views on work culture and expectations" [45]. That same engineer went on to explain that while money is a key factor with hiring these talented engineers and coders, many want non-monetary incentives such as wearing sandals and t-shirts to work, working on a nocturnal schedule, or smoking cannabis. These are all activities that would leave the more traditionalist management personnel in the Navy aghast. The Navy will need to embrace radical change in all aspects of its organizational culture to build trust with and retain the more avant-garde and creative minds in private industry.

4.6.2 Building Trust With Machines

But trusting humans isn't the only change that will be necessary; the Navy must learn to trust machines. Because DevOps requires the automation of certain processes such as testing, there must be trust between the humans responsible for the testing and the computer systems that are shouldering the automation load. Unlike human-human relationships where the trust level starts low and is gradually built over time, human-machine trust typically starts at a high level and is eroded with each error or bug that the human experiences [93]. The driving factors of the initial level of trust that humans have with automation are each human's comfort with the technology and their understanding of its functionality and processes.

This means that the TA and COTF personnel performing testing will need to be well-versed in automated testing procedures and the technology behind it. This is not the case currently as many of the TA personnel are of an older generation before the widespread daily use of advanced automated systems [15]. They will need to be provided with training to increase both their understanding of the systems and their comfort with the technology being used. There will also need to be a continuing education component to this training to ensure that the TA and COTF personnel remain aware of the latest trends in development and testing. This will help to change, as one engineer opined, the testers' "mentality which is firmly stuck in the old ways of operating in Reagan-era and 1990s" [45]. This training program must be given priority because the ability of TA and COTF personnel to trust automated processes is so critical to the certification of combat systems and the overall success of the DevOps implementation.

4.6.3 Communicating Change

Part of this readying of resources is the development and implementation of a robust communications plan. Using the communications framework in Figure 4.11, a plan can be devised that will not only prepare the Navy for the transition to DevOps but will also direct efforts during the transition to address resource shortfalls, areas of confusion, and any unforeseen challenges that arise. If this communications plan is encompassing enough, it will also help to sustain the growth and maturation of the DevOps culture and practices within the Navy.

Figure 4.11. Example communications Plan Structure. Source: [94].

This communications plan must summarize the vision and the impetus for changing from the current DAS to a DevOps system. It must include the potential peril of remaining static, which is the risk of being eclipsed by a resurgent Russia and rising China. It must also adequately and succinctly convey the desired end state of this change. That is to say that each person involved with the Navy's acquisitions process must walk away from reading each message, participating in each town hall event, and watching each video message with a firm understanding of what it is that they are working towards, how they fit into the overall plan, and what their role is within the organization. Failure to do so will cause personnel to feel as though they are facing a potential loss of their personal or group identity or that they are ill-prepared to enter into this new organizational construct [59]. This will lead to resistance to change and will torpedo change efforts in their infant stages.

Concurrent with repeating the vision and tasks needed to implement it across different mediums is the need to tailor each message to its audience. Each group and team with the Navy's Acquisitions Corps will have different concerns and needs that must be addressed. For example, TA and COTF personnel are responsible for ensuring that combat systems do exactly what they were designed to and what the DoD contractor who built them claims that they can. TA and COTF see their duty as being the last line of defense to prevent Sailors from being issued defective or impotent weapons systems. When communicating the vision, plan, and status of DevOps implementation to the TA and COTF staff, the message must be couched in terms that they can easily grasp and will allay their fears of providing Sailors with sub-par equipment. They must be made aware of the fact that continuous integration, development, and testing would allow them to work directly alongside developers in a manner that allows them both to identify and address system interface problems and operational deficiencies early, before they ever reach the hands of the Sailor.

These tailored messages must also be repeated often, sometimes daily, so that personnel never feel as though they exist within an information vacuum. Interspersed between the dissemination of these messages must also be a period for leaders to solicit feedback from their followers. As Kotter notes [60], leaders must both "listen and be listened to." This feedback allows leaders to identify weak points in their followers' understanding of the message and address any changes to the implementation plan that must be made. Soliciting the ideas and responses of followers and subordinates will also help to generate added support for DevOps implementation and reduce resistance to the changes being made.

4.7 A Quick Summary

To implement DevOps, the Navy will need to completely rethink the way that it approaches acquisitions and development. There are deeply-entrenched cultural practices and linear work processes that must be replaced with new cyclical processes and cultural norms. This new culture must be open and collaborative and championed by every echelon of leadership who communicate the need for it and help to train and guide their personnel towards the new normal. The new cyclical work processes must take the concepts of the current DAS and stretch them into infinite loops that create continuous improvement of not only the combat systems being developed but also the work processes themselves. Of these work processes, the testing and development cycle will need to be reconfigured for multiple stages of testing and a push for integration early and often, leveraging best practices for MBSE. The certification and ATO processes will also need to be rebuilt allowing for component or subsystem level certifications as well as utilizing continuous data feedback and analytics to better determine operational readiness of each combat system. This will not be fast nor will it be simple. The list of changes that must be made is captured in Table 4.1.

Table 4.1. Map of DevOps Concepts, Challenges, and Solutions

Concept	Challenges	Solutions
Open Communication and close collaboration	• Rigid organizational hierarchy • Secrecy requirements • Cultural inertia	• New organization of acquisitions personnel • Customer/System advocates • Comprehensive DevOps training
Continuous experimentation	• Statutory requirements • Rigid test processes • Rigid JCIDS process	• New continuous testing cycle • MBSE-based requirements generation • Creation of feedback loop infrastructure
Continuous feedback	• Lack of infrastructure • Secrecy requirements • Cultural inertia	• Creation of feedback loop infrastructure • Customer/System advocates • Comprehensive DevOps training
Continuous integration	• Cultural inertia • Rigid test processes • Rigid JCIDS process • Hardware requirements	• Comprehensive DevOps training • New continuous testing cycle • MBSE-based requirements generation • Use of testing sandboxes and integration testbeds
Operational flow	• Entrenched cultural practices • Rigid work processes • Rigid JCIDS process • Hardware requirements	• Champions of change • New certification processes • MBSE-based requirements generation • Use of testing sandboxes and integration testbeds

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
What Is Still Left to be Done?

In any endeavor as great as the complete revolution of the entirety of the Navy's acquisitions system, no one man or one thesis could possibly address every problem or every need of so monumental a task. Many of the solutions proposed in Chapter 4 assumed that the ideal regulatory environment existed for their implementation. In reality, this is not the case and there is a great amount of research and work that must be undertaken to solve many of the regulatory and policy challenges that face the Navy concerning the implementation of DevOps. This chapter will cover the multitude of challenges that were beyond the scope or ability of this thesis to address. These challenges will need to be surmounted by future researchers and policy makers to enable to full transformation of the Navy's acquisitions system into a DevOps system.

In any discussion of future work, it is important to keep in mind that DevOps is supposed to be a living socio-technical system that requires consistent introspection and evolution to maintain its relevance and effectiveness. It could be that the Navy realizes that DevOps is not suited for all types of systems. For instance, Ullman states [68] that Agile – and DevOps by extension – are best suited for unknown hardware systems, or new system development. For known, quantifiable hardware systems that the developers have extensive experience with and knowledge of, fasteners and circuit components for example, a traditional "waterfall" process is simpler and faster. It is also possible that the entire DevOps cycle develops over many years into something resembling the Phase-Gate Sprint Approach proposed by Dean and Van Bossuyt [95]. The Phase-Gate approach inserts quality assurance reviews into the Scrum and sprint processes to ensure that products reach a certain level of maturity before the next phase of development begins.

No matter what the future form of the DevOps acquisitions system is, there is a great deal of work that must be performed in the present to begin the shift away from "waterfall" development. Some of that work was covered in the preceding chapters but this thesis was not able to address everything. In the following sections, the remaining research and proposal work that must be undertaken will be discussed.

5.1 DevOps Contracting

One of the most difficult issues to solve is how to write contracts that adequately capture the wants and needs of the Navy and allow for the adequate amount of profit and financing for the contractor to meet those needs. As one program manager put it, "the contract is the way we communicate what we want to the contractor" [61]. To put it another way, the contract is the main vehicle through which the Navy tells the contractor what the Navy needs and what they are willing to pay for.

Therefore, any contracts written for any acquisitions or development problem must be well-tailored to the system or systems being procured. None of the professionals or experts queried during the process of developing this thesis seemed to have the answers for what a proper DevOps contract would look like or how it would function. The most typical contracts used from the Federal Acquisitions Regulation (FAR) are indefinite quantity/indefinite delivery (IDIQ) and firm-fixed-price (FFP) contracts. These have tight restrictions on how they can be implemented that don't align with the continuous and long duration of any DevOps program.

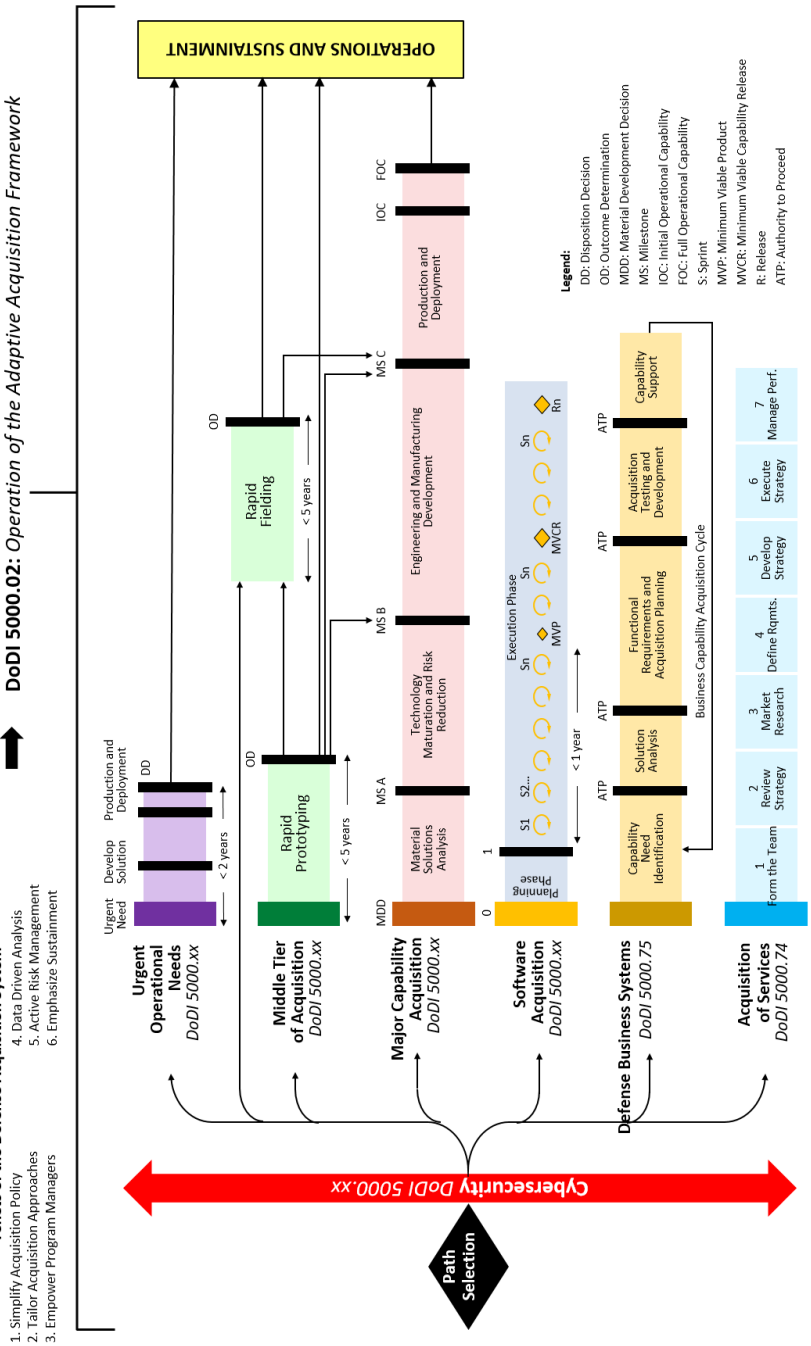

Figure 5.1. DoD Adaptive Acquisitions Framework. Source: [96].

Currently, all DevOps and Agile development programs are operating under waivers [52] or workarounds through gray areas in current acquisitions regulations [55] [50]. Many are operating under Section 804 rapid prototyping processes according to the DoD Instruction 5000.02 [97]. Section 804 programs (As shown in Figure 5.1) are allowed leeway in how they spend money and how they write contracts. While Section 804 programs are excellent for rapid development, they have a limited lifespan, requiring a working prototype within five years of development [66]. This does not address the indefinite lifespan of any DevOps program. Any future work must find a way to address the longevity of a DevOps acquisitions program as well as answer the following questions:

- How does the Navy contract out DevOps so that it can maximize value and not have to constantly rewrite the contract every time the Navy needs to add capability or make design changes?
- How does the Navy determine value of a proposal beyond lowest cost?
- How does the Navy make best value decisions when awarding contracts?

5.2 Changing Statutory Requirements for DevOps

Like the regulations of the DoD and FAR, portions of Title 10 of the U.S. Code will need to be rewritten and adapted to the realities of DevOps. Current statutory requirements in Title 10 exist to protect the government from malfeasance and fraud. This is why the requirements for testing and evaluation are so strict [51]. These protections must be left in place but adapted to the continuous testing environment and rapid change within a DevOps acquisitions system. Future work must determine who to rewrite Federal law to avoid conflicts of interest between the government and contractors, protect the government from purchasing dysfunctional systems, and ensure that all members of the military's acquisitions corps will be good stewards of the taxpayers' money.

5.3 DoD and Navy Policy Changes for DevOps

It would be impossible for any single article or thesis, no matter how long, to address the multitudinous regulations and administrative policies that will need to be adjusted to make create an environment amenable to the establishment of a DevOps system within the

Navy. These policies cover everything from the management of both uniformed military and civilian personnel to the record keeping necessary for each acquisitions and development program. These are far ranging and will require an in depth review by legal professionals and Navy administrative staffs to account for the full library of regulatory paperwork that will need to be amended. Furthermore, those same experts will need to be consulted on the proper changes that must be made in legal writing to ensure that all of the Navy's business processes will be executed within the full extent of the law.

5.4 Changing Cultural Norms

Of the hurdles noted in Section 3.1, none will be overcome overnight. Of the solutions proposed in Section 4.6, none of them will be implemented in only a single day. These will take years to implement, sometimes through trial and error. As one engineer noted, the implementation of DevOps within the Navy "is a generational shift and will take a decade or more to happen" [55]. This will take a long term plan and communications campaign from the Navy's leadership to achieve. Like the creation of the Navy's nuclear fleet had Admiral Rickover, the change to DevOps will require enthusiastic advocates for change to nurture the adoption process and provide guidance and motivation for all Naval personnel. It cannot be understated how important it is that Navy leadership and personnel approach the process of change with patience and a weather eye towards the future.

5.5 A Quick Summary

Adopting DevOps as the leading development model and system for the Navy is a gargantuan task and it was not possible to encapsulate all of the work necessary to make it a reality. The task of future researchers and staff will be to carry on from where this thesis leaves off. Their work will critical to making the decade-long task of revolutionizing the way the Navy acquires and develops combat systems a reality.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusion

It has been the attempt of this thesis to make plain the revolutionary nature of the transition from the current "waterfall" system to a DevOps system. Like the adoption of the steam engine in the 19th century or the rise aircraft carriers during World War II, the shift will completely change the way that the Navy thinks about how they development and maintain combat systems. It cannot be understated how radical the shift in culture and mindset must become among all facets of the Navy.

As shown in Chapter 3, there will be several hurdles that must be overcome during the implementation of DevOps. Not least of these is the shift in culture away from a timid, risk-averse organization to one that is forward-leaning and accepting of reasonable, quantifiable risk. There will also need to be sweeping changes in the regulations regarding defense acquisitions to allow for the radical (from the DoD's perspective) changes in development processes and practices that DevOps represents.

To overcome some of these hurdles, there will need to be reorganizations of the Navy's warfare development centers and acquisitions programs. As detailed in Section 4.1, this reorganization will be necessary to build closer working relationships between the defense contractors, PA staff, TA personnel, and operators. These closer relationships would foster improved communication and the ability of each program to better capture the wants and needs of the Sailors operating the systems. This would lead to faster development cycles, more reliable systems, and better user experiences. It would also reduce risk by relying upon a higher tempo of systems and component testing that provides better visibility of the state of systems development.

All of these factors combined would signify a fundamental shift of the Navy away from an antiquated acquisitions system that is shackled to the realities of the 20th century towards a modern system that is capable of meeting the challenges of the 21st century. DevOps acquisitions would be more responsive and able to implement newer technologies more easily than the Navy's current processes. This would ensure that the Navy and the United States would be be able to not only compete, but win against the world's great powers.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] M. Pomerlau, "DoD acquisition not broken, just slow," *C4ISRNET*, Jul 2016. Available: https://www.c4isrnet.com/c2-comms/2016/07/20/dod-acquisition-not-broken-just-slow/

[2] S. J. Freedberg Jr., "DoD 'Agile' software development still too slow: GAO," *Breaking Defense*, Jun 2020. Available: https://breakingdefense.com/2020/06/dod-agile-software-development-still-too-slow-gao/

[3] J. Kramer and T. Wagner, "Developmental test and requirements: Best practices of successful information systems using agile methods," *Defense Acquisition Research Journal: A Publication of the Defense Acquisition University*, vol. 26, no. 2, pp. p128–151, Apr 2019.

[4] R. Jack, "SecDevOps for Information Warfare," Jun 2018.

[5] United States Air Force, "Kessel Run | Code. Deploy. Win." Available: https://kesselrun.af.mil/about/

[6] W. B. Roper Jr., "Cloud One and DevSecOps," Department of the Air Force, Washington, DC, Tech. Rep., Sep 2019.

[7] T. Pierce, "The navy needs a new engine of innovation," *Proceedings*, vol. 144, no. 11, Nov 2018. Available: https://www.usni.org/magazines/proceedings/2018/november/navy-needs-new-engine-innovation

[8] E. A. Colby and A. W. Mitchell, "American, China, Russia, and the return of great-power politics," *Foreign Affairs*, Feb 2020. Available: https://www.foreignaffairs.com/articles/2019-12-10/age-great-power-competition

[9] H. W. Hendrick, "Ergonomics in organizational design and management," *Ergonomics*, vol. 34, no. 6, pp. 743–756, Jun 1991. Available: http://www.tandfonline.com/doi/abs/10.1080/00140139108967348

[10] J. M. Donnelly, "Navy routinely buys defective ships," *Roll Call*, Mar 2013. Available: https://www.rollcall.com/2019/03/20/navy-routinely-buys-defective-ships/

[11] Anonymous Program Manager, Personal Communication, Monterey, CA, Jan 2020.

[12] R. Jack, "Compile to Combat in 24 Hours Pilot (C2C24)," Naval Information Warfare Command, San Diego, CA, Tech. Rep., Aug 2020.

[13] J. L. Gibson, J. M. Ivancevich, J. H. Donnelly Jr., and R. Konopaske, *Organizations: Behavior, structure, processes*, 14th ed., P. Ducham, Ed. New York, New York, USA: McGraw-Hill Irwin, 2012.

[14] Naval Sea Systems Command, "NAVSEA Engineering and Technical Authority Manual," Washington, DC, p. 143, Jun 2011.

[15] Anonymous Acquisitions Professional, Interview, Washington, DC, May 2020.

[16] L. Gilman, M. Medin, J. Pahlka, and T. Stephens, *Software Is Never Done*, 1st ed., J. M. McQuade and R. M. Murray, Eds. Washington, DC: Defense Innovation Board, 2019. Available: https://innovation.defense.gov/software/

[17] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. Portland, OR: IT Revolution Press, 2018. Available: https://itrevolution.com/book/accelerate/

[18] Z. Banach, "What is DevSecOps," Oct 2019. Available: https://www.netsparker.com/blog/web-security/what-is-devsecops/

[19] J. W. G. Kim, J. Humble, P. Debois, *The DevOps Handbook: How to Create World-Class Agility, Reliability*, 1st ed. Portland, OR: IT Revolution Press, 2016. Available: https://itrevolution.com/book/the-devops-handbook/

[20] A. C. Ward and D. K. Sobek II, *Lean Product and Process Development*, 2nd ed. Cambridge, MA: Lean Enterprise Institute, Mar 2014. Available: https://www.leanuk.org/product/lean-product-and-process-development-2nd-edition/

[21] A. Borton, "What is Kanban?" Sep 2018. Available: https://docs.microsoft.com/en-us/azure/devops/learn/agile/what-is-kanban

[22] M. LeMay, *Agile for Everybody*, 1st ed. Sebestapol, CA: O'Reilly Media, Inc., 2019. Available: https://www.oreilly.com/library/view/agile-for-everybody/9781492033509/

[23] E. Thompson, *Agile project management : The step by step guide that you must have to learn project management correctly from the beginning to the end*, 1st ed. Seattle, WA: Amazon Publishing, Jun 2019. Available: https://www.amazon.com/Agile-Project-Management-Correctly-Beginning/dp/1072162784/ref=tmm{_}pap{_}swatch{_}0?{_}encoding=UTF8{&}qid={&}sr=

[24] Anonymous Project Manager, Personal Communication, Monterey, CA, Jan 2020.

[25] H. W. Hendrick, *Handbook of human factors and ergonomics methods*, 1st ed., N. A. Stanton, A. Hedge, K. Brookhuis, E. Salas, and H. W. Hendrick, Eds. Boca Raton, FL: CRC Press, Aug 2004. Available: https://www.researchgate.net/publication/312904128{_}Handbook{_}of{_}Human{_}Factors{_}and{_}Ergonomics{_}Methodshttps://www.taylorfrancis.com/books/9780203489925

[26] M. Shah, "Evolving test practices at Microsoft," Nov 2017. Available: https://docs.microsoft.com/en-us/azure/devops/learn/devops-at-microsoft/evolving-test-practices-microsoft

[27] M. Senapathi, J. Buchan, and H. Osman, "DevOps capabilities, practices, and challenges," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 - EASE'18*. New York, New York, USA: ACM Press, 2018, vol. Part F1377, pp. 57–67. Available: http://dl.acm.org/citation.cfm?doid=3210459.3210465

[28] Anonymous Senior Scientist, Interview, San Diego, CA, May 2020.

[29] M. Lehmann and F. E. Sandnes, "A framework for evaluating continuous microservice delivery strategies," in *ACM International Conference Proceeding Series*. New York, New York, USA: ACM Press, Mar 2017, pp. 1–9. Available: http://dl.acm.org/citation.cfm?doid=3018896.3018961

[30] L. Zhu, L. Bass, and G. Champlin-Scharff, "DevOps and its practices," *IEEE Software*, vol. 33, no. 3, pp. 32–34, May 2016. Available: http://ieeexplore.ieee.org/document/7458765/

[31] J. Maguire, "Applying DevOps practices to Pay-TV," *CED Magazine*, pp. 1–5, Oct 2014. Available: https://search-proquest-com.libproxy.nps.edu/docview/1613502336/abstract/DFD66E1189E44C8EPQ/1?accountid=12702

[32] A. Wiedemann, N. Forsgren, M. Wiesche, H. Gewald, and H. Krcmar, "Research for practice: The Devops phenomenon," *Communications of the ACM*, vol. 62, no. 8, pp. 44–49, Jul 2019. Available: https://dl.acm.org/doi/10.1145/3331138

[33] R. N. Amanchukwu, G. J. Stanley, and N. P. Ololube, "A review of leadership theories, principles and styles and their relevance to educational management," *Management*, vol. 5, no. 1, pp. 6–14, 2015. Available: http://article.sapub.org/10.5923.j.mm.20150501.02.html

[34] B. Tekniska Högskola and I. Andriushchenko, "Optimizing development performance through team composition and team culture factors in modern software development organizations," no. October, pp. 1–75, Oct 2019. Available: http://bth.diva-portal.org/smash/get/diva2:1410721/FULLTEXT02.pdf

[35] A. Mamdouh, Z. Mohammad, and A. S. Sultan, "DevOps development process awareness and adoption - The case of Saudi Arabia," *i-manager's Journal on Software Engineering*, vol. 14, no. 1, p. 21, Sep 2019. Available: http://www.imanagerpublications.com/article/16519

[36] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices," pp. 3909–3943, Apr 2017. Available: http://www.ieee.org/publications{_}standards/publications/rights/index.htmlhttp://ieeexplore.ieee.org/document/7884954/

[37] Anonymous Assistant Program Manager, Personal Communication, Washington, DC, Jun 2020.

[38] Atlassian, "What is DevOps?" 2019. Available: https://www.atlassian.com/devops

[39] Red Hat, "Understanding DevOps." Available: https://www.redhat.com/en/topics/devops

[40] I. Amazon Web Services, "What is DevOps?" 2020. Available: https://aws.amazon.com/devops/what-is-devops/

[41] Google, "What is DevOps? Research and Solutions," 2020. Available: https://cloud.google.com/devops

[42] I. NetApp, "What Is DevOps?" 2020. Available: https://www.netapp.com/us/info/what-is-devops.aspx

[43] G. Kim, K. Behr, and G. Spafford, *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*, 5th ed. Portland, OR: IT Revolution Press, 2013. Available: https://itrevolution.com/book/the-phoenix-project/http://en.wikipedia.org/wiki/The{_}Phoenix{_}Project:{_}A{_}Novel{_}About{_}IT,{_}DevOps,{_}and{_}Helping{_}Your{_}Business{_}Win

[44] M. Meyer, "Continuous integration and its tools," *IEEE Software*, vol. 31, no. 3, pp. 14–16, May 2014. Available: https://ieeexplore-ieee-org.libproxy.nps.edu/stamp/stamp.jsp?tp={&}arnumber=6802994https://ieeexplore.ieee.org/document/6802994/

[45] Anonymous Cyber Engineer, Interview, Port Hueneme, CA, Sep 2020.

[46] M. Junge, "Paperless Navy... Pshaw!" *Proceedings Magazine*, vol. 124, no. 7, p. 145, Jul 1998. Available: https://www.usni.org/magazines/proceedings/1998/july/paperless-navy-pshaw

[47] J. M. Hall, Jessika S.; O'Connor, "Learning technology adoption: Navy barriers and resistance," Thesis, Naval Postgraduate School, Monterey, CA, Mar 2018. Available: http://hdl.handle.net/10945/58306

[48] L. C. Buhl, "Mariners and machines: Resistance to technological change in the American Navy, 1865-1869," *The Journal of American History*, vol. 61, no. 3, p. 703, Dec 1974. Available: https://academic.oup.com/jah/article-lookup/doi/10.2307/1899928

[49] C. Ulsh and M. Z. Mccarty, "Vignette 2 – F22 : DevOps on a Hardware Platform," in *Software Is Never Done*, J. M. McQuade and R. M. Murray, Eds., 1st ed. Washington, DC: Defense Innovation Board, 2019, no. May, ch. Appendix 2, pp. 53–54. Available: https://innovation.defense.gov/software/

[50] Anonymous Program Manager, Interview, Patuxent, MD, Jun 2020.

[51] Anonymous Acquisitions Professional, Interview, Washington, DC, May 2020.

[52] Anonymous Program Lead, Interview, Boston, MA, May 2020.

[53] G. Boer, "What is Scrum?" Apr 2017. Available: https://docs.microsoft.com/en-us/azure/devops/learn/agile/what-is-scrum

[54] Anonymous Agile Consultant, Interview, Patuxent, MD, Jun 2020.

[55] Anonymous Chief Engineer, Interview, Monterey, CA, Jun 2020.

[56] K. H. Roberts, "Some characteristics of one type of high reliability organization," *Organization Science*, vol. 1, no. 2, pp. 160–176, May 1990. Available: http://pubsonline.informs.org/doi/abs/10.1287/orsc.1.2.160

[57] S. Shrivastava, K. Sonpar, and F. Pazzaglia, "Normal Accident Theory versus High Reliability Theory: A resolution and call for an open systems view of accidents," *Human Relations*, vol. 62, no. 9, pp. 1357–1390, Sep 2009. Available: http://www.uk.sagepub.com/http://journals.sagepub.com/doi/10.1177/0018726709339117

[58] D. van Stralen, "HRO models," 2017. Available: http://high-reliability.org/hro-models

[59] E. H. Schein and P. Schein, *Organizational Culture and Leadership*, 5th ed. Hoboken, NJ: Wiley & Sons, Inc., 2017.

[60] J. P. Kotter, *Leading change*, 2nd ed. Boston, MA: Harvard Business Review Press, 2012.

[61] Anonymous Testing Manager, Personal Communication, Monterey, CA, Feb 2020.

[62] Naval Sea Systems Command, "NAVSEA Engineering and Technical Authority Policy," p. 17, Dec 2014.

[63] J. A. Morales, H. Yasar, and A. Volkmann, "Weaving security into DevOps practices in highly regulated environments," *International Journal of Systems and Software Security and Protection*, vol. 9, no. 1, pp. 18–46, Jan 2019. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/IJSSSP.2018010102

[64] Chairman of the Joint Chiefs of Staff, "JCIDS Process Instruction," p. 114, Aug 2018. Available: http://acqnotes.com/wp-content/uploads/2018/11/CJCSI-5123.01H-Charter-of-the-Joint-Requirements-Oversight-Council-JROC-and-Implementation-of-the-JCIDS-31-Aug-2018.pdf

[65] B. D. Manning, "JCIDS Process," Feb 2020. Available: http://acqnotes.com/acqnote/acquisitions/jcids-overview

[66] B. D. Manning, "Rapid acquisitions," Apr 2020. Available: http://acqnotes.com/rapid-acquisitions

[67] C. Barrett, *Test and Evaluation Management Guide*, 5th ed., J. D. Claxton, Ed. Fort Belvoir, VA: Defense Acquisitions University, 2005, no. January. Available: https://www.dau.edu/tools/t/Test-and-Evaluation-Management-Guide-(TEMG)

[68] D. G. Ullman, *Scrum For Hardware Design*, 1st ed. Independence: David G. Ullman, 2019. Available: https://www.mechdesignprocess.com/scrum-hardware-design

[69] C. Hofmann, S. Lauber, B. Haefner, and G. Lanza, "Development of an agile development method based on Kanban for distributed part-time teams and an introduction framework," in *Procedia Manufacturing*. Amsterdam, Netherlands: Elsevier B.V., Jan 2018, vol. 23, pp. 45–50. Available: https://linkinghub.elsevier.com/retrieve/pii/S2351978918304633

[70] G. McGraw, "Risk Management Framework," in *Cybersecurity & Infrastructure Security Agency*, Jul 2017, pp. 229–270. Available: https://us-cert.cisa.gov/bsi/articles/best-practices/risk-management/risk-management-framework-(rmf)http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-5225-2503-5.ch007

[71] D. Hellem, "What is Agile development?" Oct 2017. Available: https://docs.microsoft.com/en-us/azure/devops/learn/agile/what-is-agile-development

[72] G. Schuh, S. Schroder, F. Lau, and T. Wetterney, "Next generation hardware development: Requirements and configuration options for the organization of procurement activities in the context of Agile new product development," in *2016 Portland International Conference on Management of Engineering and Technology (PICMET)*, D. Kocaoglu, Ed. Portland, OR: IEEE, Sep 2016, pp. 2583–2591. Available: http://ieeexplore.ieee.org/document/7806809/

[73] G. Schuh, F. Lau, S. Schröder, and T. Wetterney, "Next generation hardware development: The role of technology intelligence to reduce uncertainty in Agile new product development," in *PICMET 2016 - Portland International Conference on Management of Engineering and Technology: Technology Management For Social Innovation, Proceedings*, D. F. Kocaoglu, Ed. Portland, OR: IEEE, 2017, pp. 2573–2582.

[74] Anonymous Project Manager, Personal Communication, Dahlgren, VA, Jan 2020.

[75] Anonymous Assistant Program Manager, Interview, Fort Belvoir, VA, Jun 2020.

[76] Scaled Agile, "Model-Based Systems Engineering," Feb 2020. Available: https://www.scaledagileframework.com/model-based-systems-engineering/

[77] PEO Soldier, "The US Army's Adaptive Squad Architecture Initiative treats squad as integrated combat platform," Aug 2019. Available: https://soldiersystems.net/2019/08/21/the-us-armys-adaptive-squad-architecture-initiative-treats-squad-as-integrated-combat-platform/

[78] S. J. Freedberg, "Soldiers, coders surprise army brass by changing IVAS goggles," *Breaking Defense*, Dec 2019. Available: https://breakingdefense.com/2019/12/soldiers-coders-surprise-army-brass-changing-ivas-goggles/

[79] N. Garzaniti, S. Briatore, C. Fortin, and A. Golkar, "Effectiveness of the Scrum methodology for Agile development of space hardware," in *2019 IEEE Aerospace Conference*. Piscataway, NJ: IEEE, Mar 2019, vol. 2019-March, pp. 1–8. Available: https://ieeexplore.ieee.org/document/8741892/

[80] M. Laanti, "Piloting Lean-Agile hardware development," in *ACM International Conference Proceeding Series*. New York, NY, USA: ACM, May 2016, vol. 24-May-201, pp. 1–6.

[81] "Innovation boosts fleet readiness," 2019. Available: https://navylive.dodlive.mil/2019/07/15/innovation-boosts-fleet-readiness/

[82] G. L. B. Lima, G. A. L. Ferreira, O. Saotome, A. M. Da Cunha, and L. A. V. Dias, "Hardware development: Agile and co-design," in *Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015*. Las Vegas, NV: Institute of Electrical and Electronics Engineers Inc., May 2015, pp. 784–787.

[83] S. Doherty, "Chaos Monkey guide for engineers," Oct 2018. Available: https://www.gremlin.com/chaos-monkey/

[84] L. E. Hart Lockheed Martin and G. LauraEHart, "Introduction To Model-Based System Engineering (MBSE) and SysML," Lockheed Martin, Philadelphia, PA, Presentation, Jul 2015.

[85] S. Guckenheimer, "What is Monitoring?" Apr 2017. Available: https://docs.microsoft.com/en-us/azure/devops/learn/what-is-monitoring

[86] M. J. Bayer, J. M. B. O'Connor, R. S. Moultrie, and W. H. Swanson, "Cybersecurity Readiness Review," Department of the Navy, Washington, DC, Tech. Rep., Mar 2019. Available: https://www.wsj.com/public/resources/documents/CyberSecurityReview{_}03-2019.pdf?mod=article{_}inline

[87] P. Jayaram, "SQL Server replication: Overview of components and topography," Sep 2018. Available: https://www.sqlshack.com/sql-server-replication-overview-of-components-and-topography/

[88] H. Pandey, "Data replication in DBMS," Aug 2019. Available: https://www.geeksforgeeks.org/data-replication-in-dbms/

[89] Program Executive Office Integrated Weapons Systems, "Element Certification Policy," p. 8, Jul 2019. Available: http://doi.wiley.com/10.1002/j.2161-4296.1950.tb00530.x

[90] Naval Sea Systems Command, "Naval Warfare Systems Certification Policy," Washington, DC, p. 149, Aug 2012.

[91] B. D. Adams and L. E. Bruyn, "Trust in automated systems literature review," Department of National Defence, Toronto, Canada, Tech. Rep., Jun 2003. Available: https://cradpdf.drdc-rddc.gc.ca/PDFS/unc17/p520342.pdf

[92] S. Anderson, "Navy aims for "Compile to Combat in 24 Hours"," *CHIPS Magazine*, Jul 2018. Available: https://www.doncio.navy.mil/CHIPS/ArticleDetails.aspx?ID=10501

[93] K. A. Hoff and M. Bashir, "Trust in automation: Integrating empirical evidence on factors that influence trust," *Human Factors*, vol. 57, no. 3, pp. 407–434, May 2015. Available: http://www.ncbi.nlm.nih.gov/pubmed/25875432http://journals.sagepub.com/doi/10.1177/0018720814547570

[94] W. Foundation, "Workbook A: Creating a Communications Plan," 2007. Available: https://www.wallacefoundation.org/knowledge-center/Documents/Workbook-A-Communication.pdf

[95] J. H. Dean and D. L. Van Bossuyt, "Breaking the Tyranny of the Semester: A Phase-Gate Sprint Approach to Teaching Colorado School of Mines Students Important Engineering Concepts, Delivering Useful Solutions to Communities, and Working on Long Time Scale Projects," *International Journal for Service Learning in Engineering, Humanitarian Engineering and Social Entrepreneurship*, vol. 0, no. 0, pp. 222–239, Dec 2014. Available: https://ojs.library.queensu.ca/index.php/ijsle/article/view/5570

[96] Department of Defense, "Operation of the Adaptive Acquisition Framework," pp. 1–17, Jan 2020. Available: https://www.esd.whs.mil/DD/.

[97] B. D. Manning, "Middle Tier Acquisition (Section 804)," Dec 2019. Available: http://acqnotes.com/acqnote/acquisitions/middle-tier-acquisitions

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California